



# **ZX SPECTRUM**

(TS 2068)

## **Programación de juegos en lenguaje ensamblador**

**Stuart Nicholls**





# ZX SPECTRUM (TS 2068)

Programación de Juegos  
en lenguaje ensamblador

**CONSULTORES EDITORIALES  
AREA DE INFORMATICA Y COMPUTACION**

**Antonio Vaquero Sánchez**

Catedrático de Informática  
Facultad de Ciencias Físicas  
Universidad Complutense de Madrid  
ESPAÑA

**María Lourdes Fournier García**

Actuaria, Facultad Ciencias UNAM  
Profesora Asociada a Tiempo Completo  
Universidad Autónoma Metropolitana  
MEXICO

**Gerardo Quiroz Vieyra**

Ingeniero en Comunicaciones y Electrónica  
Escuela Superior de Ingeniería Mecánica y Eléctrica IPN  
Carter Wallace, S. A.  
Universidad Autónoma Metropolitana  
Docente DCSA  
MEXICO

**Alfonso Pérez Gama**

Ingeniero Electrónico  
Universidad Nacional de Colombia  
COLOMBIA

**José Portillo**

Universidad de Lima  
PERU

**Elías Lopata Szmiga**

Departamento de Ingeniería de Sistemas  
Facultad de Ingeniería  
Universidad Metropolitana —UNIMET—  
VENEZUELA



# ZX SPECTRUM (TS 2068)

## Programación de Juegos en lenguaje ensamblador

Stuart Nicholls

### **Traducción:**

Gonzalo Olivares Ruiz  
Departamento de Electrónica  
Facultad de Ciencias  
Universidad de Granada

María Jesús Vicente Pérez  
Departamento de Inglés Científico  
Universidad de Granada

### **Revisión técnica:**

Antonio Vaquero Sánchez  
Catedrático de Informática  
Facultad de Ciencias Físicas  
Universidad Complutense de Madrid

**McGraw-Hill**

MADRID • BOGOTÁ • BUENOS AIRES • GUATEMALA • LISBOA • MÉXICO  
NUEVA YORK • PANAMÁ • SAN JUAN • SANTIAGO • SAO PAULO  
AUCKLAND • HAMBURGO • JOHANNESBURGO • LONDRES • MONTREAL  
NUEVA DELHI • PARÍS • SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY  
TOKIO • TORONTO

**ZX SPECTRUM (TS 2068). Programación de Juegos en lenguaje ensamblador**

Prohibida la reproducción total o parcial de esta obra,  
por cualquier medio, sin autorización escrita del editor.

DERECHOS RESERVADOS © 1985, respecto a la primera edición en español por  
LIBROS MCGRAW-HILL DE MEXICO, S. A. DE C. V.  
Atlacomulco, 499-501, Naucalpan de Juárez, Edo. de México  
Miembro de la Cámara Nacional de la Industria Editorial, Reg. Núm. 465

ISBN: 968-451-800-5

Traducido de la primera edición en inglés de  
**Assembly Language for Arcade Games and other fast Spectrum Programs**

Copyright © 1984, por McGraw-Hill, Book Company (UK) Limited  
ISBN: 0-07-084729-0

**Edición exclusiva para Ediciones La Colina, S. A. (España)**

ISBN: 84-7615-065-2  
Depósito legal: M. 31.227-1985

Compuesto en Graflia, S. L. Pajaritos, 19. 28007 Madrid  
Impreso en Gráficas EMA. Miguel Yuste, 27. 28037 Madrid

PRINTED IN SPAIN - IMPRESO EN ESPAÑA

A mi esposa Fran por la ayuda que me prestó pasando a máquina el manuscrito original (una tarea realizada sin entender ni una palabra del tema).





# CONTENIDO

---

<b>Prefacio</b>	ix
<b>Capítulo 1</b> Imprimir con colores	1
<b>Capítulo 2</b> PLOT, DRAW, y CIRCLE	8
PLOT	8
DRAW x,y,a	20
CIRCLE x,y,r	21
<b>Capítulo 3</b> Contar	22
<b>Capítulo 4</b> Números aleatorios	30
<b>Capítulo 5</b> El teclado	38
Utilización de la variable LAST KEY	38
Utilización de IN A, (C)	39
<b>Capítulo 6</b> Movimiento	43
Movimiento de primer plano de un cuadro de carácter	43
Movimiento de fondo de un cuadro de carácter	46
Movimiento de pixel	62
<b>Capítulo 7</b> Música y efectos de sonido	70
<b>Capítulo 8</b> ATTRIBUTE, SCREEN\$ y POINT	83
ATTRIBUTE (línea, columna)	83
SCREEN\$ (línea, columna)	85
POINT	87
<b>Capítulo 9</b> La impresora	89
COPY	89
LPRINT	89
LLIST	90
<b>Capítulo 10</b> Conversión de programa	91
<b>Apéndice 1</b> Reemplazamiento de rutinas de la ROM	118
<b>Apéndice 2</b> Lista de códigos máquina de la Spectrum (Z80)	123
<b>Apéndice 3</b> Tablas de conversión decimal-hexadecimal	129
<b>Apéndice 4</b> Literales de calculadora útiles	131
<b>Apéndice 5</b> Mapa de memoria del fichero de visualización	132
<b>Apéndice 6</b> Programa de despedida	133



# PREFACIO

---

¡Usted ya ha examinado con dificultad manuales sobre programación en lenguaje máquina Z80 para la Spectrum y se le ha dado un conocimiento claro de todas las instrucciones! Pero justo cuando empiezan a ponerse interesantes, terminan con unos cuantos ejemplos de cómo sumar y restar, y entonces debe usted apañárselas por sí mismo.

Este libro empieza donde terminan ese tipo de libros y le enseña a utilizar los conocimientos que ya tiene sobre las rutinas de la Spectrum, para hacer programas de trabajo completos, escritos totalmente en código máquina.

Estudiaremos las rutinas más fáciles, tales como la impresión, y también las rutinas más complicadas como las de números aleatorios y de cálculo con números de coma flotante.

El capítulo final le enseña a aplicar la información mostrada en el libro, para convertir completamente un programa BASIC de juegos en código máquina.

Los listados en código máquina han sido desarrollado con el Ensamblador de McGraw-Hill. No voy a insistir demasiado en el hecho de que cualquiera que se tome en serio la escritura de programas en lenguaje máquina debe usar un Ensamblador, ya que de esa forma se evitará la búsqueda tediosa en tablas, asociada normalmente con el lenguaje máquina, y lo que es más importante, le ayudará a modificar fácilmente su programa cuando se descontrola (lo que sucede a menudo cuando se empieza).



# 1 IMPRIMIR CON COLORES

Aunque la impresión en código máquina es una de las tareas más aburridas y que más tiempo lleva, también es una de las más importantes. Un buen programa puede estropearse por una mala composición, o por una visualización con mensajes de entrada ambiguos, y, de igual forma, se puede mejorar un programa mediocre con una buena composición. Antes de desarrollar un programa se debe tener muy en cuenta la presentación visual.

La Spectrum tiene 21 gráficos definibles por el usuario (UDGs) para hacer un uso completo de ellos. Por ejemplo, si usted está desarrollando un juego del tipo invasores del espacio, no se detenga redefiniendo los invasores, bombas y disparos; utilice los caracteres que tiene para redefinir los números desde el cero al nueve, para presentar las puntuaciones con caracteres futuristas y para crear paisajes interesantes. Utilice los colores de forma sensata; no ponga juntos el *rojo* y el *magenta* (o el *amarillo* y el *blanco*); puede ser conveniente ejecutar su programa con un aparato en blanco y negro para comprobar si aún así se ve bien, ya que muchos propietarios de la Spectrum no usan un aparato en color.

Como cada programa requiere la impresión de información en una u otra forma, en este capítulo veremos diferentes métodos de convertir la instrucción PRINT del BASIC al código máquina.

La primera orden de un programa en BASIC se refiere normalmente a las asignaciones de las sentencias BORDER, PAPER e INK (colores principales) seguida de CLS para poner esos colores en pantalla.

El color del BORDER (margen) es el más fácil de elegir en código máquina, y sólo necesita cinco bytes. En primer lugar se carga el registro A con el "número del color" requerido y después se hace una llamada a la rutina 8859 d (229Bh) de la ROM. Esta rutina selecciona en la pantalla el margen solicitado

10 BORDER 5: REM verde

LLAMADA ESTABLECER MARGEN

org 23760  
23760 3E 05

ld a,5

23762 CD 9B 22  
23765 C9

call 8859  
ret

**Programa 1.1**

y almacena el valor del color en la variable BORDCR 23624 del sistema (de hecho el valor almacenado es ocho veces el valor del color) es decir, los bits 5-3. Los otros bits son utilizados para almacenar los parámetros INK, BRIGHT, y FLASH de la mitad inferior de la pantalla (las *líneas de entrada*).

El programa 1.1 muestra la versión en código máquina para asignación de BORDER 5 (verde).

Los colores principales para PAPER e INK se mantienen en la variable ATTR-P (23693) del sistema. Los bits 0-2 guardan los colores INK, y los bits 3-5 los colores PAPER. Los bits 6 y 7 restantes se utilizan para FLASH y BRIGHT. Para FLASH 0 el bit 7 se pone a cero; para FLASH 1 el bit 7 se pone a uno. BRIGHT 0 tiene el bit 6 a cero y BRIGHT 1 tiene el bit 6 a uno.

El programa 1.2 muestra la asignación de PAPER 3: INK 1: CLS y demuestra el hecho de que ATTR-P (23693) puede ponerse como  $IY + 83$ . El registro IY se utiliza como puntero para la variable del sistema, y contiene el valor 23610. Si desea poner "LD(23609), A" por ejemplo, esto se puede considerar como  $(IY - 1)$  o  $(IY + 255)$ .

5 REM establecer colores de pantalla permanentes	Llamada abrir canal 2	
10 PAPER 3: INK 1: CLS	3766 3E 02	ld a,2
	23768 CD 01 16	call 5633
	llamada CLS	
LD (ATTR-P), colores	23771 CD 6B 0D	call 3435
23762 FD 36 53 19 ld (iy+83), 25	23774 C9	ret

## Programa 1.2

La orden CLS (CALL 3435) borra la pantalla y establece los parámetros de pantalla, como el CLS del BASIC, *pero* observará que antes de hacer la llamada a CLS hay una llamada a la rutina 5633 de la ROM. Esto es muy importante, y se usará muchas veces en todo este Capítulo. La llamada es necesaria para abrir el canal 2, permitiendo que la información vaya a la mitad superior de la pantalla, es decir, a las líneas desde la 0 hasta la 21. Una vez establecidos nuestros parámetros permanentes de pantalla, podemos entonces pasar realmente a imprimir en la pantalla.

El programa 1.3 muestra la forma de imprimir un carácter en la pantalla en la *siguiente* posición de impresión; esta será 0,0, si se ha ejecutado una orden CLS. Observará de nuevo que el "canal 2" debe abrirse primero, y que el código del carácter a imprimir debe estar en el registro A.

Se llama a la rutina ROM para imprimir un carácter utilizando RST 16 d, que de hecho es una llamada a la rutina 5618 d. Esta rutina imprime el carácter del registro A, y avanza una posición de impresión.

REM imprimir un carácter usando colores permanentes 20 PRINT "A";	CARGAR EN AC CODIGO CAR. A
	23765 3E 41                      ld a,65
org 23760	LLAMADA IMPRIMIR
Abrir canal 2	23767 D7                      rst 16
	23768 C9                      ret
23760 3E 02                      ld a,2	
23762 CD 01 16                  call 5633	

### Programa 1.3

El programa 1.4 amplía lo anterior para permitir la impresión de una palabra "corta". De nuevo se abre el canal 2, y se carga sucesivamente el registro A con cada carácter hasta que se imprime la palabra completa.

10 REM Imprimir una serie de caracteres	E	
20 PRINT "HOLA"	23770 D7	rst 16
	23771 3E 4C	ld a,76
	L	
org 23760	23773 D7	rst 16
	23774 3E 4C	ld a,76
Abrir canal 2	L	
	23776 D7	rst 16
23760 3E 02                      ld a,2	23777 3E 4F	ld a,79
23762 CD 01 16                  call 5633	O	
23765 3E 4B                      ld a,72		
H		
23767 D7                      rst 16	23779 D7	rst 16
23768 3E 45                      ld a,69	23780 C9	ret

### Programa 1.4

Sin embargo puede ver que el método anterior supondría una pérdida de tiempo y de memoria si hubiese que imprimir una palabra extensa o frase, y sería mejor abordar el problema de forma diferente.

Como se demuestra en el programa 1.5, hay varias opciones posibles. La idea principal consiste en guardar el mensaje con DATA, y leer un carácter cada vez hasta que se imprima todo el contenido de DATA.



10 REM Imprimir una serie LARGA  
20 Print "AYUDAME"

```
org 23760
23760 3E 02      ld a,2
23762 CD 01 16   call 5633
23765 11 E4 5C   ld de, DATA
23768 01 07 00   ld bc,7
```

Longitud de serie

```
Lazo
23771 78      ld a,b
23772 B1      or c
23773 0B      dec bc
23774 C8      ret z
23775 1A      ld a,(de)
23776 13      inc de
23777 D7      rst 16
23778 18 F7   jr Lazo
DATA
defb 72 69 76 80 32 77 69
```

## Programa 1.5

En la *versión 1* el registro DE contiene la dirección del comienzo de DATA y es utilizado como puntero para cada carácter; el registro BC contiene la longitud de la serie, y se hace una llamada RST 16 para cada carácter sucesivo hasta que BC = 0.

```
org 3760
23760 3E 02      ld a,2
23762 CD 01 16   call 5633
23765 11 DF 5C   ld de, DATA
23768 01 07 00   ld bc,7
```

```
LLAMADA IMPRIMIR SERIE
23771 CD 3C 20   call 8252
23774 C9      ret
DATA
defb 72 69 76 80 32 77 69
```

La *versión 2* es igual que la versión 1 salvo en que se utiliza la rutina 8252 d de la ROM, que de hecho es una copia de la rutina de la versión 1, desde la dirección 23771 a la 23778.

```
org 23760
23760 3E 02      ld a,2
23762 CD 01 16   call 5633
23765 11 E5 5C   ld de DATA
Lazo
23768 1A      ld a, (de)
23769 CB 7F   bit 7,a
23771 20 04   jr nz, END
23773 D7      rst 16
```

```
23774 13      inc de
23775 18 F7   jr Loop
END
23777 CB BF   res 7,a
23779 D7      rst 16
23780 C9      ret
DATA
defb 72 69 76 80 32 77 197
```

La *versión 3* puede utilizarse para imprimir series de cualquier longitud, pero está limitada a los códigos de carácter del 0 al 127. El último carácter se utiliza como marca para indicar el final de la serie añadiendo 128 a su código, asignando el bit 7, y entonces, antes de imprimir un carácter se hace una prueba para ver si el bit 7 es cero. Si lo es, se repite el lazo PRINT; si no, se hace un salto a la rutina END que pone a cero el bit 7,A, imprime el carácter, y continúa con el programa, en este caso con RET.

La *versión 3* puede utilizarse como "subrutina PRINT" en un programa en el que pueden almacenarse juntos varios mensajes

con indicadores de final, y todo lo que se necesitaría sería cargar DE con la dirección de comienzo del mensaje requerido y hacer una llamada a la subrutina PRINT. Con la Spectrum se utiliza una rutina similar, aunque más complicada, para imprimir palabras clave y mensajes de error; esta rutina empieza en la dirección 3082 d.

Los mensajes de error desde 1 a R, según se muestra en el Apéndice B del manual de la Spectrum, pueden imprimirse fácilmente utilizando la orden RST 8. El byte que sigue a la instrucción RST 8 apunta al mensaje de error requerido. Por ejemplo, si selecciona el mensaje "K invalid colour" que es de hecho el vigésimo-primer mensaje de error, utilice entonces:

```
RST 8
defb 19 d (número de mensaje - 2)
```

No se necesita ninguna instrucción de retorno, ya que la ROM volverá automáticamente al BASIC después de imprimir el error.

Ahora podemos movernos o imprimir caracteres o series en una línea y columna específica, como en la instrucción PRINT AT línea, columna; "mensaje" del BASIC. De nuevo, si mira en el Apéndice A del manual de la Spectrum, verá que los códigos de carácter del 6 al 23 son códigos de control, y en particular el código 22 es el código de "AT". Ahora, afortunadamente para nosotros, la instrucción RST 16 reconocerá el código 22 como "PRIM AT" y utilizará los dos bytes siguientes de DATA como x,y; estas son las coordenadas que nos dan la posición del mensaje. El programa 1.6 muestra esto usando defb 22,3,5 para PRINT AT 3,5;

```
10 REM Imprimir serie en x,y
20 PRINT AT 3,5; "Gracias"
```

```
23768 01 0C 00
23771 CD 3C 20
23774 C9
DATA
```

```
ld bc,12
call 8252
ret
```

```
org 23760
23760 3E 02
23762 CD 01 16
23765 11 DF 5C
```

```
ld a,2
call 5633
ld de,DATA
```

```
En 3 5
defb 22 3 5
defs Gracias
```

## Programa 1.6

Puede utilizarse el mismo principio para TAB (CODE 23). Observe también que ENTER (CODE 13d) hará el mismo efecto que NEWLINE.

De modo que ahora podremos imprimir cualquier mensaje en cualquier posición de la línea 0 a la 21 utilizando los colores

principales. Todo lo que falta para completar la impresión de la pantalla normal es imprimir mensajes en colores poco comunes, BRIGHT, FLASH, OVER, e INVERSE.

El programa 1.7 muestra como se logra esto, y sigue los mismos principios del programa 1.6, en cuanto a que los códigos de control para PAPER, INK, BRIGHT, etc., se guardan en DATA con el byte que contiene el código de control siguiendo a la especificación de parámetros.

```
1000 REM Impresión en colores
1010 PRINT AT 4,7; PAPER 2; INK
4; FLASH 1; BRIGHT 1; OVER 1; IN
VERSE 1; "Puede leer esto"
```

```
org 23760
23760 3E 02      ld a,2
23762 CD 01 16   call 5633
23765 11 E2 5C   ld de, DATA
23768 01 24 00   ld bc,36
23771 CD 3C 20   call 8252
```

```
REINICIAR COLORES PERMANENTES
Y BRIGHT 0 FLASH 0
```

```
23774 CD 4D 0D      call 3405
23777 C9           ret
DATA
defb 22 4 7 17 2 16 4 18 1
defb 19 1 21 1 20 1
defs Puede leer esto
defb 21 0 20 0
```

## Programa 1.7

Después de utilizar colores secundarios es necesario poner a cero los colores principales para seguir imprimiendo. En el caso de OVER y de INVERSE se requiere utilizar RST 16, pero PAPER, INK, BRIGHT y FLASH pueden reiniciarse mediante una llamada a la rutina 3405 d de ROM que copia el valor de ATTR P (23693) en ATTR T (23695).

Finalmente, consideramos la impresión en la mitad inferior de la pantalla, es decir en las líneas de entrada 22 y 23. Esta utiliza exactamente el mismo método que el que se describió para imprimir en la parte superior, excepto que hay que abrir un canal diferente antes de utilizar RST 16. El programa 1.8 muestra un ejemplo de como se lleva esto a cabo.

```
10 REM Imprimir en líneas INPUT
20 PRINT #0 AT 0,10; "Línea input
0"; AT 1,10; "Línea input 1"
30 PAUSE 0
```

```
org 23760
```

```
Abrir canal -3
```

```
23760 3E FD      ld a,253
23762 DC 01 16   call 5633
23765 11 EA 5C   ld de, DATA
23768 01 1E 00   ld bc, 30
23771 CD 3C 20   call 8252
```

```
ESPERAR
```

```
23774 76                parar
```

```
Si pulsa tecla entonces
BIT 5, FLAGS será uno
```

```
23775 FD CB 01 6E bit 5, (iy+1)
23779 28 F9          jr z,WAIT
23781 FB CB 01 AE res 5, (iy+1)
23785 C9           ret
DATA
```

```
defb 22 0 10
defs Línea input 0
defb 22 1 10
defs Línea input 1
```

## Programa 1.8

El canal en este caso es -3 (253).

El programa en BASIC es interesante porque en el manual no se menciona que con PRINT # 0 se imprimirá en las líneas de entrada. La instrucción PAUSE 0 es necesaria para parar el programa sin la instrucción 0 O.K. que podría cambiar la impresión de la línea de entrada.

De forma similar, esta rutina en código máquina requiere el equivalente a PAUSE 0. Esto se logra examinando continuamente el bit 5 de FLAGS (23611) o (IY + 1) después de una instrucción HALT. La instrucción HALT espera a la siguiente rutina KEYSKAN, y si se pulsa una tecla, pone el bit 5 de FLAGS a uno. Así, esperamos en el lazo hasta que el bit 5 se pone a uno y entonces se vuelve al BASIC. Observar que la línea 22 se cuenta como línea 0 y la línea 23 como línea 1.

Una rutina ROM que puede ser muy útil para la impresión de las líneas de entrada, pero que deja intacta la parte superior de la pantalla, puede encontrarse en 3652, que es un "número específico de borrado de rutina de líneas"; el número de líneas se cuenta desde la parte inferior de la pantalla y debe ser mayor que 0. En el registro B se pone el número de líneas (1-24) y después se hace la llamada; los colores usados para borrar las líneas son los contenidos en ATTR P. Las líneas de entrada pueden borrarse con las instrucciones

```
LD B, 2
CALL 3652
(Return)
```

Antes de pasar al siguiente capítulo hay que destacar un punto importante referente a los gráficos definibles por el usuario. Si ha escrito un programa para ejecutarlo en máquinas de 16K y de 48K, entonces tendrá que situar correctamente los UDGs en cada máquina. La forma más fácil de hacer esto es situar los UDGs empezando a partir de la dirección 32600, es decir para una máquina de 16K, y después utilizar el Programa 1.9 para relocalizar los UDGs según la dirección guardada en la variable UDG (23675) del sistema.

```
10 REM Tranferir 16 K de UDG's para
corregir localización para máquinas de 48K
20 LET de=PEEK 23675+256*PEEK
23676
```

```
30 LET h1=32600
40 LET bc=166
50 POKE de, PEEK h1
60 LET hl=hl+1
70 LET de=de+1
80 LET bc=bc-1
```

```
90 IF bc<>0 THEN GO TO 50
100 REM volver
```

```
org 23760
23760 ED 5B 7B 5C      ld de,(23675)
23764 21 58 7F        ld hl,32600
23767 01 A8 00        ld bc,168
23770 ED B0          ldir
23772 C9             ret
```

## Programa 1.9

## 2 PLOT, DRAW, Y CIRCLE

### PLOT

La orden PLOT x,y del BASIC lleva consigo la visualización del *pixel* columna x, fila y con un color primario. El pixel de la posición 0,0 está situado en la esquina inferior izquierda de la pantalla y el 175.255 en la esquina superior derecha.

En lenguaje máquina hay dos puntos de entrada útiles a la rutina PLOT de la ROM: el primero de ellos, que es probablemente el más útil y fácil de usar es CALL 8933. Antes de llamar a la rutina, el registro B debe contener el valor de y (en el rango de 0 a 175), y el registro C debe contener el valor de x (0-255). El programa 2.1 muestra la orden PLOT 75,125 del BASIC.

```
1000 REM Plot x,y
1010 PLOT 75,175
```

```
org 23760
23760 06 7D      ld b,125
23762 0E 4B      ld c,75
23764 CD E5 22   call 8933
23767 C9        ret
```

#### Programa 2.1

Si se usan números en decimal en vez de en hexadecimal, es más fácil seguir el programa cargando los registros B y C por separado. Por supuesto, podríamos haber ahorrado memoria usando LD BC,32075 pero esto sería confuso.

El segundo punto de entrada está en 8924. En este caso se requiere que los valores de x e y se coloquen en la *pila de la calculadora*, poniéndose el valor de y en el valor superior. El programa 2.2 demuestra esto con la misma PLOT 75,125.

```
org 23760
23760 3E 4B      ld a,75
STACK 75
23762 CD 28 2D   call 11560
23765 3E 7D      ld a,125
STACK 175
23767 CD 28 2D   call 11560
LLAMADA PLOT
23770 CD DC 22   call 8924
23773 C9        ret
```

#### Programa 2.2

La rutina 11560 de la ROM se utiliza para poner el valor del registro A en la parte superior de la pila de la calculadora. Con la rutina PLOT situada en la dirección 8924 se sacan los dos valores superiores de la pila y se traza un pixel. De esta forma la pila de la calculadora queda reducida en dos valores.

El uso de la calculadora se tratará más detalladamente en el programa 2.5. Este segundo punto de entrada es útil si la calculadora ha sido utilizada para manipular una fórmula para dibujar gráficos, etc.

El programa 2.3 muestra como se realiza PLOT OVER 1; x,y.

10 REM Plot OVER 1;x,y	23765 06 28	ld b,40
20 PLOT OVER 1;25,40	23767 0E 19	ld c,25
	23769 CD E5 22	call 8933
org 23760	ESTABLECER OVER 0	
ESTABLECER OVER 1	ESTABLECER BITS 0 y 1 del FLAG P	
ESTABLECER BITS 0 y 1 del FLAG P	23772 AG	xor a
23760 3E 03	23773 FD 77 57	ld (iy+87) a
23762 FD 77 57	23776 C9	ret

### Programa 2.3

Este programa pone a uno los bits 0 y 1 de P FLAG (IY + 87) antes de hacer la llamada PLOT, y después los pone a cero. Si no se pone a cero OVER 0, entonces cualquier trazado o impresión futura se llevará a cabo como OVER 1.

PLOT INVERSE tiene la misma forma que PLOT OVER y requiere la puesta a uno de los bits 2 y 3 de P FLAG; la variable del sistema se pone al valor 12.

Usando el punto de entrada 8933 podemos escribir un programa para visualizar un carácter en cualquier lugar de la pantalla. La única limitación es que el código del carácter esté comprendido entre 32 (espacio) y 127 (copyright), es decir tiene que tener sus ocho-bytes de composición guardados en la ROM de generación de caracteres.

org 23760		Contador de byte	
Cargar L con código de CAR			
23760 2E 7F	ld 1,127		
23762 26 00	ld h,0	23772 06 08	ld b,8
		L3	Guardar punteo HL
Multiplicar x8			
23764 29	add hl,hl	23774 E5	push hl
23765 29	add hl,hl		
23766 29	add hl,hl	Contador de bit	
Sumar a (23606) para encontrar comienzo de carácter		23775 0E 08	ld c,8
		23777 7E	ld a,(hl)
23767 ED 5B 36 5C 1d de,(23606)		L2	
23771 19	add hl,de	23778 C5	push bc

Obtener posición x,y para plot		23798 C1	pop bc
23779 ED 4B 0E 5D ld bc,(XY)		23799 0D	dec c
		23800 20 E8	jr nz,12
Examinar si BIT está a uno		23802 C5	push bc
		Reinicializar PLOT a x-8, y-1	
23783 17	r1a		
23784 F5	push af		
		23803 ED 4B 0E 5D ld bc, (XY)	
23785 30 05	jr nc,L1	23807 05	dec b
		23808 3E F8	ld a,248
Si está a uno entonces PLOT		23810 81	add a,c
		23811 4F	ld c,a
23787 C5	push bc	23812 ED 43 0E 5D	ld (XY),bc
23788 CD E5 22	call 8933	23816 C1	pop bc
23791 C	pop bc	23817 E1	pop hl
L1		23818 23	inc hl
		Repetir hasta contador BYTE=0	
Mover PLOT a x+1,y		23819 10 D1	dinz L3
23792 0C	inc c	23821 C9	ret
23793 ED 43 0E 5D ld (XY),bc		XY	
23797 F1	pop af	defb 100 20	

## Programa 2.4

En el programa 2.4 puede ver que el código del carácter requerido se coloca en el registro L (dirección 23761), después se multiplica por 8 y se suma con el valor almacenado en la variable del sistema (23606) para encontrar el comienzo en el generador de caracteres de los ocho bytes que componen el carácter. El registro HL se usa después como *puntero*, conteniendo la dirección del byte del carácter. El registro BC se usa como contador para la matriz de 8 bitx8bit del carácter, y el registro A contiene el byte del carácter.

La instrucción RLA se ejecuta 8 veces para cada byte de carácter, y si un bit se pone a uno, es decir, si después de RLA la bandera CARRY se pone a uno, entonces se visualiza el pixel.

Después de cada instrucción RLA se actualizan las coordenadas x,y. Las coordenada x,y iniciales son las del pixel superior izquierdo del carácter que está siendo visualizado. Por lo tanto, para que permanezca en la pantalla, el valor de y debe estar comprendido entre 7 y 175. El valor de x puede ser de 0 a 255; si se intenta tomar un valor de x superior a 255 resultará que el valor se hace 0, es decir, si  $C = 255$  entonces "inc C" dará  $C = 0$ .



Este programa puede modificarse fácilmente para visualizar caracteres con una rotación de 90 grados en sentido contrario al de las agujas del reloj.

org 23760		23785 30 05	jr nc,L1
Cargar L con código de CAR.		Si es uno entonces PLOT	
23760 2E 60	ld 1,96	23787 C5	push bc
23762 26 00	ld, h0	23788 CD E5 22	call 8933
Multiplicar × 8		23791 C1	pop bc
23764 29	add hl,hl	L1	
23765 29	add hl,hl	Mover PLOT a x,y+1	
23766 29	add hl,hl		
Sumar a (23606) para encontrar comienzo de carácter		23792 04	inc b
23767 ED 5B 36 5C	ld de, (23606)	23793 ED 43 0E 5D	ld (XY),bc
23771 19	add hl,de	23797 F1	pop af
Contador de byte		23798 C1	pop bc
23772 06 08	ld b,8	23799 0D	dec C
L3		23800 20 E8	jr nz,L2
Guardar puntero HL		23802 C5	push bc
23774 E5	push hl	Reinicializar PLOT a x+1,y-8	
Contador de bit		23803 ED 4B 0E 5D	ld bc, (XY)
23775 0E 08	ld c,8	23807 0C	inc c
23777 7E	ld a(hl)	23808 3E FB	ld a,248
L2		23810 80	add a,b
23778 C5	push bc	23811 47	ld b,a
Obtener posición x,y de plot		23812 ED 42 0E 5D	ld (XY), bc
23779 ED 4B 0E 5D	ld bc, (XY)	23816 C1	pop bc
Examinar si BIT es uno		23817 E1	pop hl
23783	rla	23818 23	inc hl
3784 F5	push af	Repetir hasta que contador BITE=0	
		23819 10 D1	djnz L3
		23821 C9	ret
		XY	
		defb 100 20	

## Programa 2.5

En el programa 2.5 las coordenadas x,y se actualizan para visualizar un byte *vertical* en vez de uno *horizontal* como ocurría en el programa 2.4; ahora las coordenadas x,y indican el pixel inferior izquierdo, y entonces el rango para y va de 0 a 167.

Una vez estudiada la forma de visualizar un carácter podemos pasar ahora a nuestro primer programa serio para *visualizar un mensaje* en cualquier posición de la pantalla, y para que sea más útil, permitiremos la selección de su anchura y al-

tura. Se dispone de una copia del programa en cinta -LOAD "PLOTDEMO".

Para que sea útil como rutina de un programa en BASIC, la he enganchado a una rutina de "búsqueda" para que los parámetros altura/anchura, posiciones x,y de visualización, y mensaje, puedan asignarse utilizando variables del BASIC de forma que no haya necesidad de la orden POKE con códigos máquina. Si desea utilizar la rutina dentro de un programa en código máquina, la rutina de búsqueda es entonces innecesaria y puede omitirse; los parámetros pueden copiarse directamente en la memoria intermedia de la impresora. El programa BASIC es auto-explicativo y permite al lector experimentar con valores diferentes de altura/anchura, x,y y mensajes (a\$).

El código máquina es una extensión del programa 2.4. Los valores de x,y,h y w se encuentran utilizando la rutina de búsqueda. Esta rutina trabaja con el almacén de variables hasta que encuentra la variable CODE requerida, y entonces almacena el valor de la variable (0-255) en la *memoria intermedia de la impresora*. Una vez que se ha almacenado el valor de la variable, se encuentra y se almacena LEN a \$ de nuevo (0-255), y finalmente se copia el mensaje en la memoria intermedia de la impresora (el lugar más útil para almacenar datos). La rutina "PLOT" incluye una instrucción CATCHALL y convertirá cualquier parámetro equivocado en otro aceptable, y se compone de cinco bucles:

- (L1) Dibujar un bit establecido con SET para WIDTHw
- (L2) Repetir L1 para los ocho bits
- (L3) Repetir HEIGHT h veces
- (L4) Repetir para los ocho bits
- (L5) Repetir hasta el final del mensaje

Observe el comentario del programa en las variables FOR/NEXT del lazo LOOP.

En el programa 2.6 el código máquina comienza en 32393. La primera rutina, de 32393 hasta 32418, inicializa el registro BC como puntero de la memoria intermedia de la impresora y llama a la rutina SET para establecer el código de la variable situada en la dirección 23728. La rutina SET llama sucesivamente a la rutina FIND para encontrar los códigos de las variables. Entonces el valor de la variable (0-255) se almacena en la dirección de la memoria intermedia de la impresora, indicada por el registro BC. Así, las variables x,y,h, y w se localizan y almacenan consecutivamente. La rutina de 32419 a 32442 encuentra el código de la variable a\$, entonces se guarda la longitud (0-255)

## INSTRUCCIONES

Este primer programa de demostración le permitirá «dibujar» (plot) mensajes en cualquier lugar, de la pantalla usando variables BASIC para establecer los parámetros del mensaje.

Los parámetros de caracteres se establecen como sigue:

Altura	:	LET h=1 to h=22
Anchura	:	LET h=1 to h=32
eje x	:	LET x=0 to 255
eje y	:	LET y=0 to 175
Mensaje	:	LET a\$="*****"
Plot	:	RANDOMISE USR 32393

PULSE CUALQUIER TECLA

Puede seleccionar los colores de tinta antes de la llamada al código máquina PERO acuérdesese de reinicializar después el color permanente.

Cualquier parámetro incorrecto será corregido con el código/M, y si el mensaje es demasiado largo/alto se enroscará en la pantalla.

### NOTA:

NO UTILIZAR LAS VARIABLES x,y,h EN LAZOS FOR/NEXT PUES EL PROGRAMA SE ESTROPEARÁ.  
El siguiente programa le pedirá los parámetros y visualizará su mensaje.

PULSE Q PARA PARAR

```

5 OVER 0: PAPER 6: BORD ER 3: CLS
10 LET x=24: LET y=79: LET w=5: let
w=2: LET a$="PARAR LA CINTA"
15 INK 2: RANDOMIZE USR 32393
18 PAUSE 100
20 LET w=8: LET y=150: LET x=0:
LET a$="PLOT"
30 INK 1: RANDOMIZE USR 32393
35 LET a$="----": LET y=140
36 INK 1: RANDOMIZE USR 32393
40 FLASH 1: GO SUB 8000
50 FLASH 0
60 PAUSE 0
70 PAPER 7: INK 0: BORDER 7: CLS
80 LET: X=32: LET y:=175: LET h=1
: LET w=2: LET a$="INSTRUCCIONES"
90 INK 2: RANDOMIZE USR 32393:
INK 0
100 PRINT AT 2,0;"Este primer programa
de demostración le permitirá "dibujar" (plot)
mensajes en cualquier lugar de la pantalla
usando variables BASIC para establecer los
parámetros del mensaje».
110 PRINT ""Los parámetros de caracte-
res se establecen como sigue:—"
120 PRINT ""Altura: LET h = 1 to
h=22""Anchura: LET w=1 to w=3 2"
130 PRINT "eje x : LET x=0 a 255" "eje y:
LET y=0 a 175"
140 PRINT "Mensaje : LET a$=","CHR$
34: ""*****"CHR$ 34
150 PRINT "Plot : RANDOMISE USR
32393"
160 GO SUB 8000
170 IF INKEY$<>"" THEN GO TO 170
175 IF INKEY$="" THEN GO TO 175 178
CLS
180 PRINT: «Puede seleccionar los co-
lores de tinta antes de la llamada al código
máquina PERO acuérdesese de reinicializar
después el color permanente.
190 PRINT "Cualquier parámetro inco-
rrecto será corregido con el código /M, y si
el mensaje es demasiado largo/alto se en-
rroscará en la pantalla."
200 LET a$="NOTA": LET y=95: INK 2:
RANDOMIZE USR 32393: INK 0.
210 PRINT AT 12,0: "NO UTILIZAR LAS
VARIABLES x,y,h EN LAZOS FOR/NEXT
PUES EL PROGRAMA SE ESTROPEARÁ"
220 PRINT: "El siguiente programa le pe-
dirá los parámetros y visualizará su men-
saje."
230 GO SUB 8000
235 IF INKEY$<>"" THEN GO TO 235.
240 IF INKEY$="" THEN GO TO 240
242 CLS . BORDER 4
245 PLOT 0,121: DRAW 255,0: DRAW
0,-73: DRAW -255,0: DRAW 0,73
250 PRINT AT 8,0: "ALTURA 1-22 ? --",
"ANCHURA 1-32? --", "X
eje x 0-255 ? --", "eje y 0-1
75 ? __MENSAJE ? __",
-----
TINTA
0-9 ?»,,,

```

```

255 LET z$=""
"
257 LET v$=""
"
260 PRINT OVER 1,AT 8,0;z$
262 INPUT h: PRINT AT ",18,h; " "
: OVER 1;AT 8,0:v$
265 PRINT OVER 1;AT 9,0;z$: INPUT w:
PRINT AT 9,18; w; " "; OVER1 : AT 9,0: v$
270 PRINT OVER 1;AT 10,0;z$: INPUT x:
PRINT AT 10,18; x; " "; OVER1: AT 10,0:v$
272 PRINT OVER 1: AT 11,0;z$: INPUT y:
PRINT AT 11,18; y; " "; OVER 1: AT 11,0: v$
273 PRINT OVER 1;AT 12,0;z$:AT
13,0;z$: INPUT a$: PRINT AT 13,0: a$: FOR
k=LEN a$ TO 31: PRINT " ": NEXT k:
PRINT OVER 1;AT 12,0: v$: OVER 1;AT
13,0: v$
275 PRINT OVER 1;AT 14,0;z$: IN PUT i:
PRINT AT 14,0: INK i; OVER1; v$
280 PAUSE 50
290 CLS : INK 1: RANDOMIZE USR 3
2393: INK 0
300 PRINT #0;AT 0,0: "PULSE Q PARA PA-
RAR": AT 1,0: "CUALQUIER OTRA PARA IR
OTRA VEZ"
310 PAUSE 0: IF INKEY$="q" OR IN
KEY$="Q" THEN STOP
320 CLS : GO TO 245
8000 LET h=1: LET w=2: LET x=24.
LET y=15: LET a$="PULSE CUALQUIER
TECLA"
INK 2: RANDOMIZE USR 32393: INK 0
8010 RETURN
9000 STOP
9800 CLEAR 32334: LOAD ""CODE: GO
TO 5
9900 SAVE "PLOTDEMO" LINE 9800
9950 SAVE "LARGE" CODE 32335,265
org 23760 32335
FIND
32335 2A 4B 5C Id, hl, (23627)
L2
32338 3A B0 5C Id a,(23728)
32341 BE cp (hl)
32342 CB ret z
32343 CB 6E bit 5,(hl)
32345 20 08 jr nz,L1
32347 23 inc hl
32348 5E Id e,(hl)
32349 23 inc hl
32350 56 Id d,(hl)
32351 19 add hl,de
32352 23 inc hl
32353 18 EF jr L2
L1
32355 CB 76 bit 6,(hl)
32357 20 0C jr nz,L3
32359 23 inc hl
32360 7E Id a,(hl)
32361 CB 7F bit 7,a
32363 28 FA jr z,-6
32365 11 06 00 Id de,6
32368 19 add hl,de
32369 18 DF jr L2

```

L3			
32371 CB 7E	bit 7,(hl)	LP3	
32373 28 F6	jr z,-10	32841 7E	ld a,(hl)
32375 11 13 00	ld, de,19	32482 E5	push hl
32378 19	add hl,de	32483 C5	push bc
32379 18 D5	jr L2	32484 06 08	ld b,8
SET		LP2	
32381 32 B0 5C	ld (23728),a	32486 C5	push bc
32384 CD 4F 7E	call FIND	32487 17	rla
32387 23	inc hl	32488 F5	push af
32388 23	inc hl	32489 DA F9 7E	jp c,PLOT
32389 23	inc hl	32492 2A 03 5B	ld hl,(23299)
32390 7E	ld a, (hl)	32495 3A B0 5C	ld a,(23728)
32391 02	ld (bc),a	32498 85	add a,l
32392 C9	ret	32499 32 B0 5C	ld (23728),a
EMPEZAR			
32393 01 00 5B	ld bc,23296	32502 C3 0F 7F	jp SKIP
32396 3E 78	ld a,120	PLOT	
32398 CD 7D 7E	call SET	32505 ED 4B B0 5C	ld Bc,(23728)
32401 03	inc bc	LP1	
32402 3E 79	ld a,121	32509 C5	push bc
32404 CD 7D 7E	call SET	32510 ED 4B B0 5C	ld bc,(23728)
32407 03	inc bc	32514 C5	push bc
32408 3E 68	ld a,104	32515 CD E5 22	call 8933
32410 CD 7D 7E	call SET	32518 C1	pop bc
32413 03	inc bc	32519 0C	inc c
32414 3E 77	ld a,119	32520 ED 43 B0 5C	ld (23728),bc
32416 CD 7D 7E	call SET	32524 C1	pop bc
32419 3E 41	ld a,65	32525 10 EE	djnz LP1
32421 32 B0 5C	ld (23728),a	SKIP	
32424 CD 4F 7E	call FIND	32527 F1	pop af
32427 23	inc hl	32528 C1	pop bc
32428 5E	ld d,(hl)	32529 10 D3	djnz LP2
32429 23	inc hl	32531 3A 00 5B	ld a,(23296)
32430 56	ld,(hl)	32534 21 B0 5C	ld hl,23728
32431 ED 53 04 5B	ld (23300),de	32537 77	ld (hl),a
		32538 23	inc hl
32435 D5	push de		
32436 C1	pop bc	32539 AF	xor a
32437 23	inc hl	32540 7E	ld a,(hl)
32438 11 05 5B	ld de, 23301	32541 DE B0	sbc a,176
32441 ED B0	ldir	32543 38 03	jr c,+3
32443 2A 00 5B	ld hl, (23296)	32545 77	ld (hl),a
32446 AF	xor a	32546 18 08	jr +8
32447 7C	ld a,h	32548 7E	ld a,(hl)
32448 DE B0	sbc a,176	32549 FE 00	cp 0
32450 38 04	jr c,+4	32551 20 02	jr nz,+2
32452 67	ld h,a	32553 36 B0	ld (hl),176
32453 22 00 5B	ld (23296),hl	32555 35	dec (hl)
32456 22 B0 5C	ld (23728),hl	32556 C1	pop bc
32459 21 05 5B	ld hl,23301	32557 E1	pop hl
LP5		32558 10 B1	djnz LP3
32462 E5	push hl	32560 23	inc hl
32463 7E	ld a,(hl)	32561 C1	pop bc
32464 26 00	ld h,0	32562 10 AB	djnz LP4
32466 6F	ld l,a	32564 3A 03 5B	ld a,(23299)
32467 29	add hl,hl	32567 87	add a,a
32468 29	add hl,hl	32568 87	add a,a
32469 29	add hl,hl	32569 87	add a,a
32470 11 00 3C	ld, de, 15360	32570 6F	ld l,a
32473 19	add hl,de	32571 3A B0 5C	ld a,(23728)
32474 06 08	ld b,8	32574 85	add a,l
LP4		32575 32 00 5B	ld (23296),a
32476 C5	push bc	32578 32 B0 5C	ld (23728), a
32477 ED 4B 01 5B	ld bc (23297)	32581 3A 01 5B	ld a,(23297)

32584 32 B1 5C	ld (23729),a	32592 3D	dec a
32587 E1	pop hl	32593 C8	ret z
32588 23	inc hl	32594 32 04 5B	ld (23300),a
32589 3A 04 5B	ld a,(23300)	32597 C3 CE 7E	jp LP5

## Programa 2.6

y finalmente se copia la serie. La rutina de 32443 a 32458 examina el valor de y (si es mayor que 175 se cambia a y - 176) y después almacena los valores x e y anteriores en 23296/7 y también en 23728/9.

El código máquina restante es una extensión de la rutina plot (dibujar un carácter) con los bucles adicionales para visualizar con PLOT el pixel "w (anchura)" veces para "h líneas"; se hace una prueba después de cada "línea" para asegurarse de que la siguiente posición PLOT permanece en la pantalla. La rutina desde 32540 a 32554 comprueba la siguiente posición de PLOT y origina un efecto de enrosque si está fuera de la parte superior de la pantalla.

Sin profundizar ahora demasiado en las rutinas CALCULATOR (de la calculadora) podemos estudiar un programa que hace uso de PLOT en el punto de entrada 8924.

El programa 2.7 muestra como puede dibujarse una onda sinusoidal utilizando CALCULATOR para manejar la fórmula:

$$88 + 80 * \text{SEN} (A/128 * \text{PI})$$

```
1000 FOR A=0 TO 255
1010 PLOT A,88+80*SIN (A/128*PI)
1020 NEXT A
```

```
org 23760
LET A=0
```

```
23760 AF      xor a
LAZO
GUARDAR A
23761 F5      push af
APILAR A
23762 CD 28 2D call 11560
23765 3E 58   ld a,88
23767 CD 28 2D call 11560
23770 3E 50   ld a,80
23772 CD 28 2D call 11560
23775 F1      pop af
23776 F5      push af
23777 CD 28 2D call 11560
23780 3E 80   ld a,128
23782 CD 28 2D call 11560
```

```
APILAR 88 80 A 128
```

USAR CALCULADORA

```
23785 EF      rst 40
Dividir 2 valores superiores (A/128)
deff 5
Stack PI/2
deb 163
Duplicar valor superior
deff 49
Sumar 2 valores superiores (PI)
deff 15
Multiplicar 2 valores superiores
(A/128*PI)
deff 4
SEN valor superior SEN (A/128*PI)
deff 31
Mutplicar 2 valores superiores
80*SIN (A/128*PI)
deff 4
Sumar 2 valores superiores
88+80*SIN (A/128*PI)
deff 15
Fin de rutina calculadora
Los 2 valores superiores son
A y 88+80*SIN (A/128*PI)
```

deff 56  
 LLAMADA PLOT  
 23795 CD DC 22  
 Obtener valor A  
 23798 F1

call 8924  
 pop af

23799 3C  
 Examinar si no cero (256)  
 23800 FE 00  
 23802 20 D5  
 23804 C9

inc a  
 cp 0  
 jr nz, LOOP  
 ret

## Programa 2.7

La rutina CALCULATOR puede utilizarse para realizar operaciones aritméticas complejas mediante el uso de *literales*. En realidad son llamadas ROM para realizar cálculos en los cuales intervienen los dos valores situados en la parte superior de la pila de la calculadora, pero pueden usarse también para manejar series. Por ejemplo, si los dos valores de la parte superior fueran 1234 y 2, y el Literal 4 fuese llamado desde la rutina CALCULATOR, entonces esos dos valores podrían sacarse de la pila, multiplicarse y devolver su producto a la parte superior de la pila; no obstante la pila se reduciría en un valor, y el valor superior sería 2468.

Los números *enteros positivos* pueden colocarse en la pila por medio de:

STACK VAL A – CALL 11560 OR STACK VAL BC –  
 CALL – 11563.

En el programa 2.7 he utilizado STACK VAL A para ordenar “variable A”, 88,80 “variable A”, y 128.

Después se llama a la rutina USE calculator mediante la instrucción RST 40. La rutina CALCULATOR opera con los bytes que siguen a RST 40 y actúa sobre cada uno, llamando a otras subrutinas según sea necesario. En nuestro programa el siguiente byte es 5, y Literal 5 significa “GOSUB DIVIDE”. Entonces se toman los dos valores superiores de la pila, se divide el primero por el segundo, y se devuelve el resultado a la pila. El siguiente byte de la rutina es Literal 163; le indica a la calculadora “STACK PI/2” (PILA PI/2).

- El literal 49 es DUPLICATE TOP VALUE (duplicar el valor superior).
- El literal 15 es ADD TOP TWO VALUES (sumarle dos al valor superior) así se apila la suma.
- El literal 4 es MULTIPLY TOP TWO VALUES (multiplicar por dos el valor superior).
- El literal 32 es SINE OF TOP VALUE (seno del valor superior) de esta forma se reemplaza el valor superior por su seno.

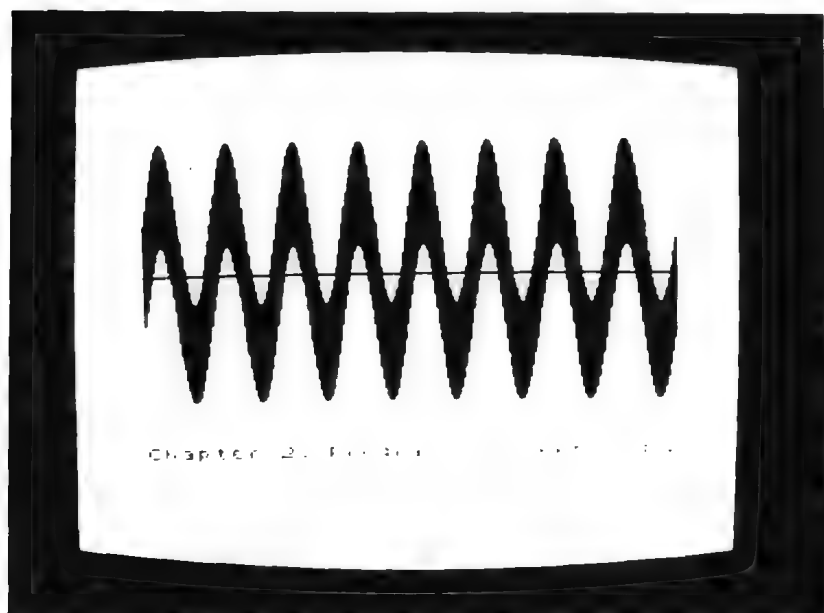


Hay muchos más literales que pueden utilizarse en programas de juegos y serán tratados en capítulos posteriores.

Por último, el Literal 56 es "END CALCULATOR ROUTINE CALL" (llamada a la rutina fin de calculadora) que *debe* utilizarse para volver a su programa en código máquina. Otro punto importante a tener en cuenta cuando se utiliza la rutina CALCULATOR es que, antes de volver al BASIC deben borrarse todas las variables apiladas previamente.

Al final de la rutina CALCULATOR quedan dos valores en la pila de la calculadora; el valor superior es  $88 + 80 * \text{SEN}(A/128 * \text{PI})$  y el otro es el valor de "A". La llamada PLOT (8924) toma entonces esos dos valores, borra la pila, y dibuja (PLOT) el pixel. Después se incrementa el valor "A" y se repite LOOP hasta que "A" = 256 (es decir, "A" se pone a 0).

Con este programa puede ver que los literales de CALCULATOR son muy útiles y que una fórmula complicada se ha resuelto con sólo ocho bytes. En realidad necesita más bytes para utilizar la pila que para realizar los cálculos.



## DRAW

Para la orden DRAW x,y hay también dos puntos de entrada. El primer punto (call 9402) requiere que el valor ABS de x esté en el registro C y que el valor ABS de y esté en el registro B. Se utiliza el registro DE para guardar el SGN de x e y; el registro D contiene SGNx (1 si es positivo, 255 si es negativo) y el registro E guarda SGNy. El programa 2.8 es una versión modificada del programa 2.7 para hacer una demostración de la orden DRAW.

5 PLOT 0,95: DRAW 255,0		23789 3E 78	ld a,120
10 FOR a=0 TO 255		23791 CD 28 2D	call 11560
20 PLOT a,120+40*SIN (a/16*PI)		23794 3E 28	ld a,40
DRAW 0,-50		23796 CD 28 2D	call 11560
30 NEXT a		23799 F1	pop af
		23800 F5	push af
		23801 CD 28 2D	call 11560
		23804 3E 10	ld a,16
		23806 CD 28 2D	call 11560
		23809 EF	rst 40
		defb 5 163 49 15 4 31 4 15 56	
org 23760		23189 CD DC 22	call 8924
Plot 0,95		Stoe H'L'	
23760 06 5F	ld b,95	23822 D9	exx
23762 0E 00	ld c,0	23823 E5	push hl
23764 CD 1E5, 22	call 8933	23824 D9	exx
Guardar H'L'		DRAW 0,-50	
23767 D9	exx	23825 06 32	ld b,50
23768 E5	push hl		
23769 D9	exx	23827 0E 00	ld c,0
Draw 255,0		23829 16 FF	ld d,255
23770 06 00	ld b,0	23831 1E 01	ld e,1
23772 0E FF	ld c,255	23833 CD BA 24	call 9402
23774 16 01	ld d,1	Restore H'L'	
23776 1E 01	ld e,1	23836 D9	exx
23778 CD BA 24	call 9402	23837 E1	pop hl
		23838 D9	exx
Recuperar H'L'		23839 F1	pop af
23781 D9	exx	23840 3C	inc a
23782 E1	pop hl	23841 FE 00	cp 0
23783 D9	exx	23843 20 C4	jr nz, L1
23784 AF	xor a	23845 C9	ret
L1			
23785 F5	push af		
23786 CD 28 2D	call 11560		

## Programa 2.8

Observe que antes de llamar a DRAW el registro HL debe ser almacenado (en el ejemplo se hace guardándolo en la pila) porque es utilizado en la rutina DRAW.

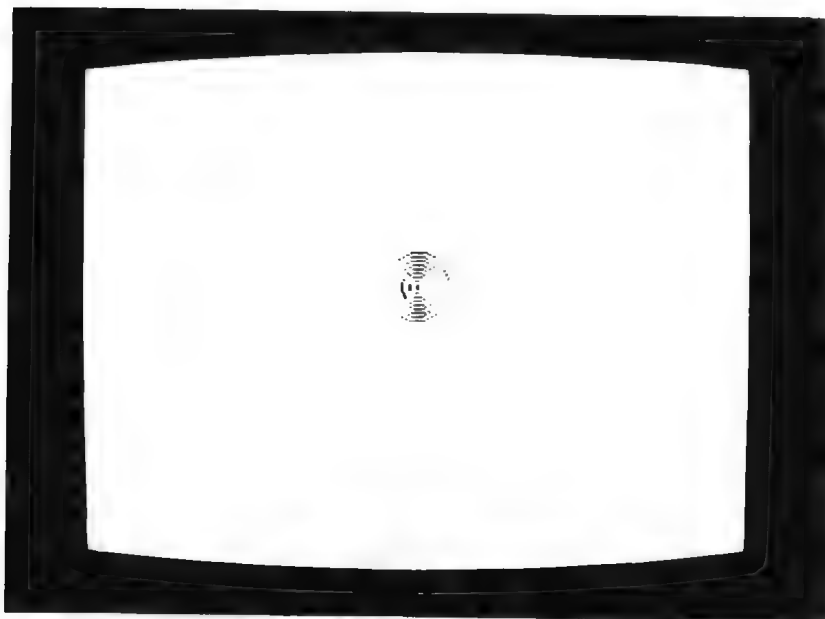
El valor del registro HL no debe ser modificado durante una rutina en código máquina escrita por un usuario o el programa se descontrollará al regresar al BASIC. El valor del registro fundamental HL es recuperado después de la orden DRAW.

**DRAW OVER** e **INVERSE** se utilizan de la misma forma que **PLOT OVER/INVERSE**.

El segundo punto de entrada para **DRAW x,y** requiere que los dos valores superiores de la pila de la calculadora contengan **x** e **y**; de nuevo, **y** es el valor superior -CALL 9335. Si desea utilizar este segundo punto de entrada necesitará leer el Capítulo 7 para tomar contacto con el método de la Spectrum de almacenar números *negativos* de cinco bytes en la pila de la calculadora.

## **DRAW x,y,a**

El punto de entrada para esta orden BASIC es 9108 y requiere que los valores de **x,y** y **a** estén colocados en la pila de la calculadora (en ese mismo orden, es decir, "**a**" en la parte superior). De nuevo, en el Capítulo 7 se explicará como se apilan números negativos. Al igual que antes, tampoco debe cambiarse el valor del registro fundamental HL.



## CIRCLE x,y,r

El punto de entrada para esta orden del BASIC es 9005 y requiere que los valores x,y, y r estén en la parte superior de la pila de la calculadora. El registro fundamental HL se guarda durante la llamada CIRCLE. El programa 2.9 muestra una forma de dibujar círculos concéntricos para conseguir un efecto de rizo.

10 REM Circle x,y,r		23780 F1	pop af
20 FOR B=1 TO 2		23781 F5	push af
30 FOR A=1 TO 21 STEP 2		23782 CD 28 2D	call 11560
40 CIRCLE OVER 1,128,88,A		Guardar H'L' (IMPORTANTE!)	
50 NEXT A			
60 NEXT B;		23785 D9	exx
		23786 E5	push hl
		23787 D9	exx
org 23760			
For B=1 TO 2		Call CIRCLE	
23760 06 02	ld, b2	23788 CD 2D 23	call 9005
L2			
23762 05	push bc	Recuperar H'L'	
For A=1 TO 21		23791 D9	exx
23763 3E 01	ld a,1	23792 E1	pop hl
L1		23793 D9	exx
23765 F5	push af		
Set OVER 1		23794 F1	pop af
		STEP 2	
23766 FD 36 57 03 ld (iy+87),3		23795 3C	inc a
		23796 3C	inc a
Stack DATA		23797 FE 17	cp 23
		23799 20 DC	jr nz, L1
23770 3E 80	ld a,128	23801 C1	pop bc
		23802 10 D6	djnz L2
		Reinicializar OVER 0	
23772 CD 28 2D	call 11560		
23775 3E 58	ld a,88	23804 FD 36 57 00	ld (iy+87),0
23777 CD 28 2D	call 11560	23808 C9	ret

## Programa 2.9

Observará que el programa es casi tan lento como el BASIC. Es porque la rutina CIRCLE de la ROM es bastante larga. Si su programa en código máquina necesita un círculo, es más rápido guardar los puntos del dibujo como DATA y usar un lazo para dibujar cada pixel.

# 3 CONTAR

La mayoría de los programas de juegos y muchos otros “serios” requieren alguna forma de contar. Esto se utiliza mucho en programas de juegos para llevar la puntuación, vidas perdidas, tiempo empleado, etc. Conozco tres métodos para contar y visualizar el resultado en la pantalla. El primero consume tanta memoria y es tan confuso, que sólo daremos una breve descripción. Este método lleva consigo la asignación del número MAX permitido con DATA. Por ejemplo, si usted estuviera contando desde 0 a 999999, entonces se reservarían seis bytes de DATA y se comenzaría por 000000. Después se necesitaría una rutina de impresión de series para imprimir los seis ceros en la pantalla. Cuando se incrementa en uno el contador será necesario empezar por los bytes de las unidades, examinar que no es nueve, sumar uno a las unidades y después imprimir la serie. Si las unidades llegan al valor nueve se pondrían a cero, después la columna de las decenas se examina para ver si es nueve y así sucesivamente. Después de imprimir la serie se debe hacer una comprobación para ver si los seis bytes son iguales a nueve, y si es así se termina de contar. Si se quiere contar hacia adelante o hacia atrás en pasos mayores que uno hay que colocar el contador “incrementar en uno” en un lazo.

El segundo método se utiliza para contar hacia adelante o hacia atrás en un rango comprendido entre 0 y 65535 como se muestra en el programa 3.1.

org 23760  
Poner contador a cero

23760 01 00 00      1d bc,0  
L1  
23763 ED 43 B0 5C      1d (23728), bc

Apilar valor

23767 CD 2B 2D      call 11563

Abrir canal 2

23770 3 E 02      1d a, 2  
23772 CD 01 16      call 5633

Establecer PRINT AT 11,16

23775 3E 16      ld a,22  
23777 D7      rst 16

23778 3 E 0B      ld a,11  
23780 D7      rst 16

23781 3E 10      ld a,16  
23783 D7      rst 16

Llamada IMPRIMIR VALOR EN PILA

23784 CD E3 2D      call 11747

Obtener valor y sumar 1  
 23787 ED 4B B0 5C    ld bc, (23728)  
 23791 03            inc bc

Volver si es cero  
 23792 78            ld a,b  
 23793 B1            or c  
 23794 C8            ret z

Examinar si pulsa tecla  
 23795 FD CB 01 6E bit 5, (iy+1)

23799 28 DA            jr z,L1  
 23801 C9            ret

### Programa 3.1

En este programa se utiliza una rutina ROM que imprime el valor que hay en la parte superior de la pila de la calculadora: call 11747. Los bytes que sobran en las variables 23728/9 del sistema son utilizados para guardar el valor del contador; el rango va desde 0 a 65535. Todo lo que necesitamos hacer después es coger el valor actual y sumarle uno, poner el nuevo valor otra vez en COUNTER (contador), apilar (STACK) el valor, hacer la asignación de parámetros con PRINT AT y llamar a PRINT VALUE ON STACK (imprimir el valor de la pila). Si el valor de BC se hace cero (es decir, 65.536) se vuelve al BASIC.

He incluido también una rutina EXIT situada en las direcciones desde 23795 a 23800 de forma que pulsando una tecla se volverá también al BASIC. Es aconsejable utilizarla, ya que de otra forma nos quedaríamos bloqueados en LOOP hasta que el contador llegue a 65535 y eso podría llevar varios minutos.

La tercera forma de contar utiliza la calculadora. Hasta ahora hemos considerado a la pila de la calculadora como un lugar para almacenar valores, sin explicar con detalle como se almacena. Los valores están realmente almacenados en la pila en forma de cinco-bytes; es decir, toman cinco bytes de información para definir cualquier número admitido por la computadora. En el Capítulo 7 se discute esto con más detalle. En el programa 3.2 hacemos uso de este hecho para contar desde 0 en adelante en pasos de 250 sin límite superior. Sin embargo, una vez que se llega a 99999999 se acude a la rutina PRINT STACK VALUE para imprimir el valor en su forma E (1E + 9 etc.), pero probablemente un valor de cien millones es suficientemente alto para la puntuación de cualquier juego.

org 23760  
 Utilizar CALCULADORA

23760 EF            rst 40

LITERAL STACK 0  
 defb 160

Terminar de usar CALCULADORA  
 defb 56

Establecer HL a STACKEND-5  
 después de instrucción defb 56

L1  
 Copiar valor en MEMORIA INTERMEDIA DE IMPRESORA

23763 11 00 5B            ld de,23296  
 23766 01 05 00            ld bc,5  
 23769 ED B0            ldir

Abrir canal 2

23771 3E 02           ld a,2  
23773 CD 01 16       call 5633

Establecer PRINT AT 11,16

23776 3E 16           ld a,22  
23778 D7              rst 16  
23779 3E 0B           ld a,11  
23781 D7              rst 16  
23782 3E 10           ld a,16  
23784 D7              rst 16

Llamada PRINT VALUE ON STACK

23785 CD E3 2D       call 11747

STACK 250

23788 3E FA           ld a,250  
23790 CD 28 2D       call 11560

APILAR VALOR

23793 21 00 5B       ld hl,23296  
23796 ED 5B 65 5C    ld de,(23653)  
23800 01 05 00       ld bc,5  
23803 ED B0          ldir  
23805 ED 53 65 5C    ld (23653),de

Usar CALCULADORA

23809 EF              rst 40

SUMAR LITERAL

defb 15  
Terminar de usar CALCULADORA  
defb 56

Examinar si se pulsa tecla

23812 FD CB 01 6E bit 5,(iy+1)

23816 28 C9           jr z,L1

Sacar último valor de PILA

23818 2A 65 5C       1d hl,(23653)  
23821 2B              dec hl  
23822 2B              dec hl  
23823 2B              dec hl  
23824 2B              dec hl  
23825 2B              dec hl  
23826 22 65 5C       ld (23653),hl  
23829 C9              ret

## Programa 3.2

El programa tiene varias rutinas importantes que requieren explicación. La asignación del valor 0 se realiza en la pila de la calculadora mediante defb 160, que es el literal STACK 0; los cinco bytes de la parte superior de la fila son después tomados y copiados en la memoria intermedia de la impresora para utilizarse como almacén. Esto se lleva a cabo con una instrucción LDIR; se asigna el valor de HL para nosotros con la rutina de la calculadora (STACKEND-5). El valor inicial se imprime en 11, 16 con la rutina PRINT VALUE y así se borra la pila.

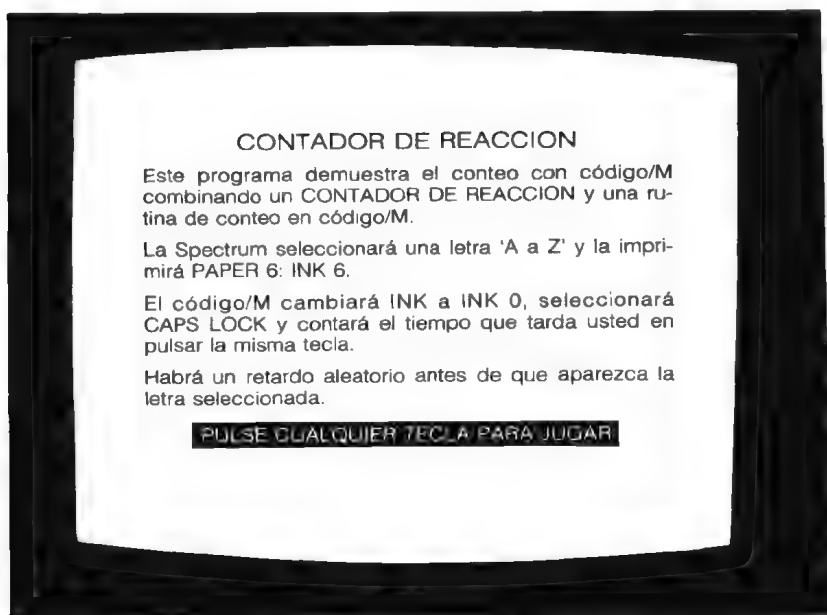
Lo mismo con la rutina ADD 250. Requiere que el valor de 250 esté colocado en la pila por medio de STACK VAL A, y después se pone en la pila el valor "almacenado". Esto se realiza utilizando LDIR. El comienzo del valor almacenado se pone en el registro HL; el valor de STACKEND se coloca en DE. Después de la instrucción LDIR debe devolverse a la variable del sistema la nueva dirección de STACKEND. La calculadora se utiliza para sumar estos dos valores y para repetir las rutinas STORE y PRINT VALUE. De nuevo se añade una rutina de pulsación de tecla para permitir la salida de la cuenta, pero observe que la pila se reinicializará antes de volver al BASIC.



La cuenta hacia atrás requiere que el valor de comienzo se coloque en la pila (ver Capítulo 7 para números mayores que 65535) y que se use el Literal 3 SUBTRACT. Observe que en la resta, el valor "superior" es sacado desde el inferior. Al final de cada cuenta se requiere una prueba para ver si los bytes del "almacén" están todos a cero y, si es así, finaliza la cuenta (o si el segundo byte tiene el bit 7 a 1, el número es negativo). En la cuenta atrás es también necesario borrar la visualización del número anterior, antes de imprimir el nuevo, porque si se hace un cambio desde, por ejemplo, 1000 a 999, el "cero de las unidades" de 1000 podría permanecer en la pantalla, y el número que aparecería sería 9990.

El programa COUNTDEMO, disponible en cinta, muestra como utiliza la rutina de contador para contar el tiempo de reacción.

El programa BASIC es auto-explicativo y utiliza principalmente las instrucciones de impresión. Se utilizan sentencias DATA para guardar los parámetros de x,y,h,w y a\$ para la impresión LARGE, según se trata en el Capítulo 2.





```

12>PAPER 6:CLS
15 DATA 87,143,1,2,"CONTAR"
20 DATA 95,63,1,2,"LENTO"
30 DATA 71,63,1,2,"NORMAL"
40 DATA 95,63,1,2,"BUENO"
50 DATA 63,63,1,2,"MUY BUENO"
60 DATA 63,63,1,2,"EXCELENTE"
80 DATA 47,63,1,2,"NINGUN INTENTO"
90 DATA 0,31,1,2,"OTRA VEZ? s/n"
95 DATA 55,125,5,2,"GRACIAS"
96 DATA 103,79,3,2,"por"
97 DATA 71,50,5,2,"JUGAR"
100 DATA 24,79,5,2,"PARE LA CINTA"
105 DATA 15,167,1,2,"TIEMPO DE REAC-
CION"
110 DATA 15,164,2,2,"-----"
----"
120 DATA 24,15,1,2,"PULSE CUALQUIER
TECLA"
125 RESTORE 100
130 FOR a=1 TO 4
140 PAUSE 25
160 INK 2: GO SUB 9000
170 NEXT a
180 PAUSE 0: BORDER 5: PAPER 6.
INK 0: CLS
190 RESTORE 105
220 INK 0: GO SUB 9000
250 PRINT AT 3,0;"Este programa demues-
tra el conteo con código/M combinando un
CONTADOR DE REACCION y una rutina de
conteo en código/M"
260 PRINT "La Spectrum seleccionará una
letra (A a Z) y la imprimirá PAPER 6: INK 6."

```

270 PRINT "El código/M cambiará INK a INK0, seleccionará CAPS LOCK y contará el tiempo que tarda usted en pulsar la misma tecla."

280 PRINT "Habrá un retardo aleatorio antes de que aparezca la tecla seleccionada."

290 PRINT AT 20,5;"PULSE CUALQUIER TECLA PARA JUGAR"

300 IF INKEY\$=<>" THEN GO TO 300

302 IF INKEY\$="" THEN GO TO 302

305 CLS

310 RESTORE 105: INK 1: GO SUB 9000

320 INK 5: GO SUB 9000

330 RESTORE 15: INK 2: GO SUB 9000

350 INK 0: PLOT 102,114: DRAW 50

0: DRAW 0-38: DRAW-50,0. DRAW 0,38

380 LET z=INT (RND\*26+65)

390 LET x=111: LET y=111: LET h=

4: LET w=4: LET a\$=CHR\$ z

400 INK 6: LET c=USR 32393

410 FOR a=1 TO RND\*200+50: NEXT a

420 INK 0

430 LET c=USR 23760

440 LET a=PEEK 23728+256\*PEEK 23

729

450 IF a>=140 OR a=0 THEN RESTO

RE 80

460 IF a<140 THEN RESTORE 20

470 IF a<120 THEN RESTORE 30

480 IF a<100 THEN RESTORE 40

490 IF a<85 THEN RESTORE 50

500 IF a<70 THEN RESTORE 60

510 GO SUB 9000

7000 RESTORE 90: INK 1: GO SUB 9000

7010 IF INKEY\$=<>" THEN GO TO 7010

7020 IF INKEY\$="" THEN GO TO 7020

7030 LET d\$=INKEY\$

7040 IF d\$=<>"y" THEN GO TO 8000

7050 LET c=USR 23842: GO TO 350

8000 RESTORE 95

8005 POKE 23843,20

8010 LET c=USR 23842

8015 POKE 23843,18

8030 FOR a=1 TO 3

8040 GO SUB 9000

8050 NEXT a

8060 STOP

9000 READ x,y,h,w,a\$

9010 LET c=USR 32393

9020 RETURN

9800 CLEAR 32334. LOAD " "CODE : G  
O TO 5

9900 SAVE "REACTION" LINE 9800

9950 SAVE "LARGE"CODE 32335,265

10 REM >z? \ STEP }~ CLEAR THE

N 0 OVER !?X~ RETURN 6 ?600: RETU

RN ¿ NEXT ??? OR STEP +->? STEP

? BEEP >

BEEP >?xLEN (? CLEAR THEN ?n ( 0

UT \! \PEEK VERIFY GO SUB CVAL

\ CLEAR THEN 0o<>

Org 23760

Encontrar variable z

23760 3E 7A	ld a,122
Guardar en 23681	
23762 01 81 5C	ld bc,23681
23765 CD 7D 7E	call 32381

Establecer CAPS LOCK  
23768 FD CB 30 DE set 3,(iy+48)

Intercambiar atributo de color

23772 21 00 58	ld hl,22528
L1	
23775 7E	ld a,(hl)
23776 FE 36	cp 54
PAPER 6 INK 6	
23778 20 02	jr nz,L2
23780 36 30	ld (hl),48
Reponer PAPER 6 INK 0	
L2	
23782 23	inc hl
23783 7C	ld a,h
23784 FE 5B	cp 91
Fin de ATRIBUTOS	?
23786 20 F3	jr nz L1

Empezar la cuenta

23788 01 00 00	ld bc,0
Guardar cuenta en pila	
L3	
23791 C5	push bc
Poner cuenta en CALCULADORA	
23792 CD 2B 2D	call 11563

Establecer PRINT AT 6,14  
23795 3E 02 ld a,2  
23797 CD 01 16 call 5633  
23800 3E 16 ld a,22

23802 D7	rst 16
23803 3E 06	ld a,6
23805 D7	rst 16
23806 3E 0E	ld a,14
23808 D7	rst 16

Imprimir cuenta  
23809 CD E3 2D call 11747

Obtener cuenta y sumarle 1  
23812 C1 pop bc  
23813 03 inc bc

Examinar si cuenta < 65535  
23814 78 ld a,b  
23815 B1 or c  
23816 28 0F jr z,END

Examinar tecla pulsada

23818 FD CB 01 6E bit 5, (iy+1)

23822 28 DF	jr z,L3
Encontrar INKEY\$	
23824 3A 08 5C	ld a,(23560)

```

Comparar a$
23827 21 81 5C      ld hl,23681
23830 BE            cp (hl)
23831 20 D6          jr nz,L3
END
Guardar cuenta
23833 ED 43 B0 5C ld,(23728),bc
Recuperar minúsculas
23837 FD CB 30 9E res 3, (iy+48)
23841 C9            ret
Borrar 18 líneas inferiores
23842 06 12         ld b,18
23844 CD 44 0E      call 3652
23847 C9            ret

```

### Programa 3.3

# 4 NUMEROS ALEATORIOS

Los números aleatorios son una parte esencial de la mayoría de los programas de juegos, tal como el retardo aleatorio y la elección aleatoria de letras del programa REACTION TIME (programa 3.3) del Capítulo 3. La asignación de un número aleatorio en código máquina es realmente muy fácil; puede abordarse de dos maneras, dependiendo del “azar/extensión” del número.

El primer método, que realmente no produce un número aleatorio verdadero, pero que es lo suficientemente bueno para muchos juegos, consiste en utilizar las variables FRAMES y SEED de forma distinta a como se usan normalmente. Es decir, si la extensión del número aleatorio es 0-1, 0-3, 0-7, 0-15, 0-31, 0-63, 0-127, 0-255 este método es válido. La rutina exige que el registro HL se cargue con el valor de SEED (23670); el registro DE se debe cargar con el byte alto de SEED/ y el byte bajo de FRAMES (23671); se suman los valores de los dos registros y se almacena la suma de nuevo en la variable SEED para el siguiente “número aleatorio”. El registro A se carga después con el H o el L y se realiza un “AND a” para enmascarar los bits no deseados; de ahí la extensión de número aleatorio indicada anteriormente. Por ejemplo, el programa 4.1 nos da un ejemplo de INT (RND \* 32) enmascarando los bits 7-5 del valor contenido en el registro A para conseguir números de 0 a 31.

org 23760		Set PRINT AT b,10	
10 números aleatorios		23778 3E 02	ld a,2
23760 06 0A	ld b,10	23780 CD 01 16	call 5633
L1		23783 3E 16	ld a,22
23762 C5	push bc	23785 D7	rst 16
Load HL,(SEED)		23786 78	ld a,b
23763 2A 76 5C	ld hl,(23670)	23787 D7	rst 16
23766 ED 5B 77 5C	ld de,(23671)	23788 3E 0A	ld a,10
23770 19	add hl,de	23790 D7	rst 16
SEED modificado para siguiente número		Obtener número aleatorio y ponerlo en	
23771 22 76 5C	ld (23670),hl	la PILA CALCULADORA	
Enmascarar BITS 7-5		23791 F1	pop af
23774 7D	ld a,1	23792 CD 28 2D	call 11560
23775 E6 1F	and 31	23795 CD E3 2D	call 11747
Guardar número aleatorio		Imprimirlo	
23777 F5	push af	Obtener cuenta de números	
		23798 C1	pop bc

```

repetir hasta 10 números
23799 10 D9      djnz L1
23801 C9          ret

```

## Programa 4.1

PROGRAMA 1 EJEMPLO DE SALIDA IMPRESA

```

6
22
23
8
9
25
24
5
1
10

```

Como se dijo anteriormente esto es útil si se requiere un “número aleatorio” de esta extensión. Es un poco más complicado obtener  $\text{INT}(\text{RND} * 23)$ , por ejemplo, mediante:

```

LD A, H
AND 15
LD H, A
LD A, L
AND 7
ADD A, H

```

Habrás observado que en el programa 4.1 el registro B se ha utilizado para guardar la cuenta del bucle (diez veces) y que este valor también se utiliza para la rutina `PRINT AT b, 10;`. El ejemplo impreso muestra lo aleatorio de la rutina.

Para producir un número aleatorio de la misma forma que con la orden  $\text{INT}(\text{RND} * n)$  del BASIC necesitaremos utilizar la rutina `RND` de la ROM, pero desgraciadamente esto no se puede hacer con una instrucción de llamada a `RND` ya que la rutina no vuelve después de ser ejecutada. La rutina ocupa las direcciones desde 9725 hasta 9765 de la ROM, y emplea la calculadora una vez más, para modificar el valor de `SEED`. Comienza extrayendo el valor existente de `SEED`, y termina con el nuevo valor duplicado en la parte superior de la pila. Se cambia el valor superior y se utiliza para actualizar `SEED`. Se modifica el valor que queda para dar un valor de 0 a 1 (pero no 1) que se utiliza para `RND`. Para utilizar esta rutina es necesario copiarla en RAM, y acceder a ella desde su programa como una subrutina. El programa 4.2 le muestra este método utilizado para obtener  $\text{INT}(\text{RND} * 12345)$  e imprimir 44 respuestas utilizando `CHR$ CODE 6` para que la impresión se haga en dos columnas.

INT (RND*12345)		defb 49	
		Fin RUTINA CALC	
		defb 56	
		Poner valor superior en BC	
org 23760		REGISTRO (INT 0-65535)	
CLS		23797 CD A2 2D	call 11682
23760 3E 02	ld a,2		
23762 CD 01 16	call 5633		
23765 CD 6B 0D	call 3435		
44 números aleatorios		Guardar nuevo SEED	
		23800 ED 43 76 5C ld (23670),bc	
		Manipular BYTE EXPONENTE	
		23804 7E	ld a,(hl)
		23805 A7	and a
23768 06 2C	ld b,44	23806 28 03	jr z,L2
L1		23808 D6 10	sub 16
23770 C5	push bc	23810 77	ld (hl),a
		L2	
COPIA de ROM 9725 a 9765		PILA 12345	
		23811 01 39 30	ld bc,12345
23771 ED 4B 76 5C ld bc, (23670)		23814 CD 2B 2D	call 11563
23775 CD 2B 2D	call 11563	23817 EF	rst 40
		*	
23778 EF	rst 40	defb 4	
PILA (1)		LITERAL 'ENTERO'	
defb 161		defb 39	
		Fin RUTINA CALC	
		defb 56	
		Imprimir valor de PILA	
+		23821 3E 02	ld a,2
defb 15			
PILA (75)			
defb 52 55 22			
*			
defb 4		23823 CD 01 16	call 5633
PILA (65537)		23826 CD E3 2D	call 11747
defb 52 128 65 0 0 128		PRINT ,	
LITERALES		23829 3E 06	ld a,6
defb 50 2		23831 D7	rst 16
PILA (1)			
defb 161		Obtener cuenta	
-			
defb 3		23832 C1	pop bc
Duplicar		23833 10 BF	djnz L1
		23835 C9	ret

## Programa 4.2

EJEMPLO DE SALIDA IMPRESA DEL PROG. ANTERIOR

1857	3558
7666	7091
1037	3758
10296	6834
6415	12044
3058	7178
7727	11711
3057	7150
2940	10693

La rutina RND utiliza algunos literales nuevos que pueden servir también para otras rutinas:

- El LITERAL 161 es STACK NUMBER 1.



- El LITERAL 52 es STACK DATA (es normalmente un número expresado en forma reducida – ver Capítulo 7).
- El LITERAL 50 es n-mod-m.
- El LITERAL 2 es DELETE (el cociente).

Los dos últimos literales son de poca utilidad para programas de juegos. La rutina —CALL 11682— es utilizada por la ROM para recuperar el valor superior de la pila y colocar el valor entero en el registro BC. En este caso el valor entero se redondea por defecto o por exceso, según sea necesario, para obtener el número más próximo. Hay también una rutina similar —CALL 11733— que coloca en el registro A el valor superior de forma semejante. La rutina ROM copiada acaba en 23810.

La rutina desde 23811 a 23820 modifica el valor RND de la pila. Se apila el valor 12345 y después se multiplica por el valor RND. Después se utiliza un nuevo literal, el 39; éste obtiene cualquier valor decimal del resultado de redondear *por defecto* el valor de 12345 RND (como en el RND del BASIC), y así deja  $INT(RND * 12345)$  en la parte superior de la pila. Después se emplea la rutina PRINT VALUE para imprimir 44 valores en dos columnas, y los resultados muestran que estos valores son realmente aleatorios.

El programa de demostración 4.3 muestra como se utiliza realmente RND. En este caso se utiliza RND para imprimir rascacielos de altura al azar, con caracteres y colores aleatorios, y podría utilizarse en programas del tipo “bombas sobre la ciudad” (city bomb). Todos los UDGs para un programa completo han sido ya asignados y puede que desee desarrollar este programa. El programa “RNDDEMO” de la cinta utiliza el programa BASIC para conseguir la primera visualización de pantalla; esta se mantiene en la pantalla durante unos segundos, y después se utiliza la rutina en código máquina para imprimir la siguiente visualización, mostrando la velocidad del Código Máquina y también la función RND.

Un punto a considerar cuando se usa la instrucción RND del BASIC con llamadas USR en código máquina es que debe utilizar LET “variable” = USR “dirección” para llamar al código máquina, ya que RANDOMIZE USR “dirección” afectará al generador de números aleatorios. Puede utilizar también PRINT USR dirección, si no le importa que el valor contenido en el registro BC se imprima al volver al BASIC. En la rutina en código máquina, PAPER y los colores INK se establecen utilizando LD (IY + 83), 47 en vez de LD A,47: LD (IY + 83) A, como en ejemplos anteriores, ahorrando así memoria y mostrando la utilidad del registro IY como puntero. Esto se de-

muestra posteriormente con más detalle en el programa cuando IY se pone a 23296, comienzo de la memoria intermedia de la impresora, y es utilizado para LD (23296), 22 (AT) y LD (23299), 16 (INK). Sin embargo, cuando se altera el valor contenido en IY hay que tener cuidado de que si se hace una llamada ROM que emplee el registro IY, el valor debe reinicializarse a 23610 antes de hacer la llamada, o la rutina ROM se descontrolará y se puede originar una caída.

Antes de llamar a la rutina de número aleatorio se carga el registro A con el número  $RND * n$  y éste se almacena después en la dirección 23681 (una dirección no utilizada por las variables del sistema). Este número es después recogido en la rutina RND y utilizado para obtener  $INT(RND * n)$ , el valor que se coloca en el registro A, llamando a 11733 antes de regresar de la rutina. La secuencia intermedia de la impresora es utilizada para guardar la información de PRINT en orden: AT (22), x, y, INK (16), b\$, a\$.

```

1 REM >? PASO EL" BORRAR 65/>?
PASO?
10 BORDER 0: PAPER 5: INK 7: C L5
20 FOR c=31 TO 0 STEP -1
30 LET i=19-INT(RND*10)
40 LET a$=CHR$(INT(RND*2+148))
45 LET b$=CHR$(INT(RND*4))
50 PRINT AT L-1,c;"▲"
60 FOR a=i TO 20
70 PRINT AT a,c;CHR$(16+b$+a$)
80 NEXT a
90 PRINT AT a,c;"■"
100 NEXT c
110 LET x=0: LET y=0: LET a$=""
130 PRINT AT x,y;a$
140 PRINT #0;"■ COMBUSTIBLE 10000 ■"
BOMBAS
100 ■ DISPAROS 100 ■"
150 RETURN
500 DATA 24,79,5,2,"PARE LA CINTA"
510 DATA 15, 167, 1, 2, "NUMEROS ALEATO-
RIOS"
520 DATA 15,164,1,2,""
525 DATA 24,15,1,2,"PULSE CUALQUIER TÉ-
CLA"
530 DATA 47,164,2,4,"BASIC"
540 DATA 31,164,3,2,"CODIGO MAQUINA"
600 BORDER 4: PAPER 6: INK 0: C L5
605 RESTORE 500
610 FOR a=1 TO 4
620 INK a: GO SUB 8000
630 PAUSE 50: NEXT a
640 PAUSE 0: CLS
645 PRINT INK 0;AT 8,0;"Este programa
muestra la rutina RND en código Máquina y
compara la velocidad de las versiones BASIC y
código Máquina.
650 RESTORE 525

```

```

655 PAUSE 100
660 FLASH 1: INK 2: PAPER 7: GO
SUB 8000: FLASH 0
670 PAUSE 0: CLS
1000 LET z=1
1010 GO SUB 2000
1020 GO SUB 10
1025 LET z=z+1
1030 GO SUB 2000
1040 GO SUB 3000
1045 LET z=z+1
1050 GO TO 1010
2000 FOR a=1 TO 400: NEXT a: CLS

2010 PAPER 8: INK 0: RESTORE
530+(10*(z/2=INT (z/2))): GO SUB 8000
2020 FOR a=1 TO 200: NEXT a: RETURN
3000 LET c=USR 23760: GO TO 110
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 32334: LOAD "Udg"CODE USR
"a": LOAD ""CODE: GO TO 500
9999 SAVE "RNDDEMO" LINE 9998: S
AVE "Udg"CODE USR "a",168: SAVE
"LARGE"CODE 32335,265

```

org 23760

BORDER 0

```

23760 3E 00      ld a,0
23762 CD 9B 22   call 8859

```

PAPER 5 INK 7

23765 FD 36 53 2F ld (iy+83),47

CLS

```

23769 3E 02      ld a,2
23771 CD 01 16   call 5633
23774 CD 6B 0D   call 3435

```

Establecer IY como PUNTERO

23777 FD 21 00 5B ld iy,23296

23781 FD 36 00 16 ld (iy+0),22

23785 FD 36 03 10 ld (iy+3),16

Reinicializar IY

23789 FD 21 3A 5C ld iy,23610

Contador de columnas

```

23793 3E 1F      ld a,31
L1
23795 32 02 5B   ld (23298),a

```

19-INT (RND\*10)

```

23798 3E 0A      ld a,10
23800 32 81 5C   ld (23681),a

```

23803 CD 61 5D	call RND
23806 6F	ld l,a
23807 3E 13	ld a,19
23809 95	sub 1
23810 32 01 5B	ld (23297),a

INT (RND\*2+148)

23813 3E 02	ld a,2
23815 32 81 5C	ld (23681),a
23818 CD 61 5D	call RND
23821 C6 94	add a,148
23823 32 05 5B	ld (23301),a

INT (RND\*4)

23826 3E 04	ld a,4
23828 32 81 5C	ld (23681),a
23831 CD 61 5D	call RND
23834 32 04 5B	ld (23300),a

PRINT AT x-1,y CHR\$ 147

23837 3E 02	ld a,2
23839 CD 01 16	call 5633
23842 3E 16	ld a,22
23844 D7	rst 16
23845 3A 01 5B	ld a,(23297)
23848 3D	dec a
23849 D7	rst 16
23850 3A 02 5B	ld a,(23298)
23853 D7	rst 16
23854 3E 93	ld a,147
23856 D7	rst 16
L2	

FOR a=x TO 20

PRINT AT a,y CHR\$	16+b\$+a\$
23857 11 00 5B	ld de,23296
23860 01 06 00	ld bc,6
23863 CD 3C 20	call 8252
23866 3A 01 5B	ld a, (23297)
23869 3C	inc a
23870 32 01 5B	ld (23297),a
23870 32 01 5B	ld (23297),a
23873 FE 15	cp 21
NEXT a	
23875 20 EC	jr nz,L2

PRINT AT a,y CHR\$ 153

23877 3E 99	ld a,153
23879 32 05 5B	ld (23301),a
23882 3E 07	ld a,7
23884 32 04 5B	ld (23300),a
23887 11 00 5B	ld de,23296
23890 01 06 00	ld bc,6
23893 CD 3C 20	call 8252
23896 3A 02 5B	ld a,(23298)
23899 3D	dec a
23900 FE FF	cp 255

#### SIGUIENTE COLUMNA

23902 20 93 jr nz,L1  
23904 C9 ret

#### RUTINA RND DE ROM

##### RND

23905 ED 4B 76 5C ld bc, (23670)  
23909 CD 2B 2D call 11563  
23912 EF rst 40  
defb 161 15 52 55 22 4 52 128  
defb 65 00 128 50 2 161 3 49 56  
23931 CD A2 2D call 11682  
23934 ED 43 76 5C ld (23670),bc  
23938 7E ld, a, Chl)  
23939 A7 and a  
23940 28 03 jr z,L3  
23942 D6 10 sub 16  
23944 77 ld (hl),a

INT (RND\*PEEK (23681))

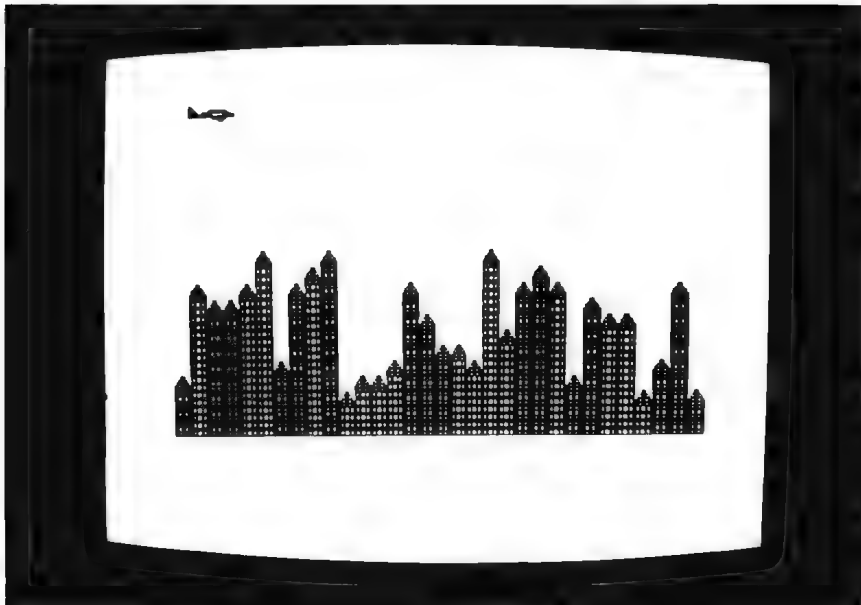
##### L3

23945 3A 81 5C ld a,(23681)  
23948 CD 28 2D call 11560  
23951 EF rst 40  
defb 4 39 56

#### VALOR en A

23955 CD D5 2D call 11733  
23958 C9 ret

### Programa 4.3



# 5 EL TECLADO

Hay dos métodos para leer el teclado en código máquina. Ambos son útiles, pero el método elegido depende del tipo de programa.

## Utilización de la variable LAST KEY

La ROM de la Spectrum contiene una rutina KEYSCAN y DECODE a la cual se accede 50 veces durante cada segundo. La rutina también actualiza el contador FRAMES.

El código de carácter de la última tecla pulsada es almacenado en la variable LAST KEY (23560) y si es una nueva tecla, entonces el bit 5, (IY + 1) se pone a uno. Incidentalmente la orden HALT espera a que se haga la siguiente llamada a KEYSCAN antes de continuar con la rutina en código máquina.

Con la anterior información podemos escribir el equivalente de PAUSE 0.

```
org 23760
```

```
ESPERAR A QUE SE PULSE TECLA
```

```
L1
```

```
23760 FD CB 01 6E bit 5,(iy+1)
```

```
23764 28 FA          jr z,L1
```

```
23766 FD CB 01 AE res 5(iy+1)
```

```
CONTINUAR CON PROGRAMA
```

```
23770 C9          ret
```

## Programa 5.1

En el programa 5.1 puede ver que se examina constantemente el bit 5, (IY + 1) para ver cuando es uno, y sólo cuando esto ocurre (indicando una nueva pulsación de tecla) el programa continúa. Observar que después de ello hay que poner a cero el bit 5 (IY + 1), ya que en caso contrario si se utilizase una rutina PAUSE similar antes de la siguiente llamada KEYSCAN, el bit 5 podría estar a uno y pasarse por alto la rutina PAUSE.

## Utilización de IN A, (C)

Es idéntica a la orden IN del BASIC leyendo la mitad de la fila especificada por el valor que contiene el registro BC, y se trata en el manual de la Spectrum. El valor que queda en A indica qué teclas se pulsán de esa mitad de fila. Cada mitad de fila contiene cinco teclas, y los bits 4-0 de A contienen el estado de esas teclas (los bits 7-5 están normalmente a uno), el bit 0 se utiliza para indicar el estado de la “tecla externa” en la mitad de fila, y el bit 4 el estado de la “tecla interna”.

Cuando no se pulsa ninguna tecla todos los bits están a uno, y A contendrá 255. Si se pulsa una tecla su bit se pondrá a cero. Por ejemplo, si se pulsase la tecla 0 entonces el bit 0 se pondría a cero y A podría valer 254. Puede ver que con este método tenemos la posibilidad de leer más de una tecla a la vez, cuando se pulsán varias al mismo tiempo, examinando el valor de A o los bits individuales de A. Esto podría ser útil en un juego para dos o más jugadores o para mover cosas en dos direcciones (por ejemplo hacia arriba y hacia abajo) al mismo tiempo. El programa 5.2 muestra la orden IN utilizada para esperar hasta que se pulse la tecla 0.

org 23760

LEER MITAD FILA 6-0

L1	
23760 10 FE EF	ld, bc, 61438
23763 ED 78	in a, (c)

EXAMINAR SI TECLA 0 ES PULSADA

23765 CB47	bit 0, a
23767 C8	ret z
23768 18 F6	jr L1

### Programa 5.2

La llamada a KEYSCAN de la ROM se hace 50 veces durante cada segundo incluso *durante nuestro propio programa en código máquina* y, por ello, el programa es más lento. Si necesitamos una velocidad extra, por ejemplo, para la visualización enrollada de pixel, podemos parar esta interrupción de nuestro programa utilizando la instrucción DI. Sin embargo, esto significa que mientras esté deshabilitada la interrupción no podemos utilizar la rutina LAST KEY para leer el teclado (a menos que hagamos nosotros la llamada a la rutina ROM) y debemos recurrir a la instrucción IN. El programa 5.3 muestra como se utiliza la instrucción DI.

org 23760		EXAMINAR SI TECLA 0 ES PULSADA
DESHABILITAR		23766 CB 47      bit 0,a
		23768 20 F7      jr nz,L1
23760 F3	di	
LEER MITAD FILA 6-0		HABILITAR
L1		23770 FB      ei
23761 01 FE EF	ld bc,61438	23771C9      ret
23764 ED 78	in a,(c)	

### Programa 5.3

Observe que la instrucción EI debe utilizarse antes de volver al BASIC, o bien quedará inutilizado el teclado. Intente probar el programa sin la instrucción EI y vea qué sucede. No se producirá ningún daño, pero la Spectrum tendrá que desenchufarse y volverse a conectar para recuperar la función del teclado.

El programa 5.4 muestra la forma de obtener PAUSE n para conseguir un retardo de tiempo 0, como en el BASIC; continúa si se pulsa una tecla. El programa utiliza el hecho de que el examen de tecla ocurre cada 0,02 segundos y de que HALT espera a que se examine el teclado.

org 23760		23768 20 05      jr nz,L2
Establecer RETARDO 10 SEG		DISMINUIR CONTADOR
		23770 0B      dec bc
23760 01 F4 01	ld bc,500	23771 78      ld a,b
L1		23772 B1      or c
LAZO DE PAUSA		23773 20 F4      jr nz, L1
23763 76	halt	L2
EXAMINAR SI SE PULSA TECLA		23775 FD CB 01 AE      res 5,(iy+1)
		23779 C9      ret
23764 FD CB 01 6E bit 5,(iy+1)		

### Programa 5.4

El programa 5.5 se encuentra en la cinta "KEYDEMO" y hace uso del método LAST KEY de lectura del teclado para producir un programa de mecanografía simple que permite el borrado y la modificación de lo escrito.

La posición de impresión actual es indicada por un cursor parpadeante. Observe que CAPS SHIFT y "2" mueve la posición de impresión a la siguiente PRINT COMMA TAB; CAPS SHIFT y "5" produce un espacio hacia atrás (pero observe el error de ROM por el cual no se puede volver un espacio desde el comienzo de la línea 1 hasta el final de línea 0), y ENTER produce un salto de línea. La tecla BREAK es inoperante en código máquina por lo que he utilizado "STOP" para volver al BASIC.



## INSTRUCCIONES

Este programa demuestra el método de lectura del teclado en código/M utilizando la variable LAST KEY y examinando el BIT 5 de FLAGS para ver si es una NUEVA tecla.

El cursor puede volver atrás usando la tecla 5 de cursor para corregir o borrar cuando se escribe.

ENTER le dará una nueva línea, y como la tecla BREAK le da un SPACE, utilice 'STOP' para volver al BASIC.

El código /M usa 43 bytes.

PULSE CUALQUIER TECLA

Como ocurre con la instrucción INKEY\$ del BASIC no se leen ni las teclas de doble-cambio ni las teclas de modo gráfico. Si desea examinar o imprimir teclas con doble-cambio, debe establecer una tabla DATA de caracteres sin cambio y de sus equivalentes con doble-cambio. Entonces si LDA (LAST KEY) regresa con el valor 14 en A, indicando que se han pulsado dos teclas con cambio, se necesitará una rutina para leer la siguiente tecla pulsada y convertir su carácter sin cambio en su correspondiente con cambio utilizando la tabla DATA.

```
10 REM >? STEP ?
20>DATA 24,79,5,2, "PARE LA CINTA"
30 DATA 7,170,2,3,"MAQUINA DE ES-
CRIBIR"
40 DATA 7,150,2,3, "-----"
50 DATA 24, 15, 1, 2"PULSE CUAL-
QUIER TECLA"
```

```
60 DATA 39,170,1,2,"INSTRUCCIONES"
65 RESTORE
70 BORDER 4: PAPER 6: CLS
80 FOR A=1 TO 4
85 PAUSE 20
90 INK A: PAPER 8: GO SUB 8000
100 NEXT A
110 PAUSE 0: PAPER 7: CLS
120 INK 2: PAPER 8: 60 SUB 8000
130 INK 0: PRINT AT 3,0; "Este programa
demuestra el método de lectura del teclado
en código/M utilizando la variable LAST KEY
```

y examinando el BIT'5 de FLAGS para ver si es una NUEVA tecla.

140 PRINT "El cursor puede volver atrás usando la tecla 5 del cursor para corregir o borrar cuando se escribe."

150 PRINT "ENTER le dará una nueva línea y como la tecla BREAK le da un SPACE, use 'STOP' para volver al BASIC."

160 PRINT "El código /M usa 43 bytes"

200 RESTORE 50: INK 2: GO SUB 8000

210 PAUSE 0: BRIGHT 1: CLS

220 INK 0: RANDOMIZE USR 23760

230 STOP

8000 READ x,y,h,w,a\$

8010 RANDOMIZE USR 32393

8020 RETURN

9990 CLEAR 32334: LOAD ""CODE : GO TO 20

9999 SAVE "LARGE" CODE 32335,265

org 23760

Abrir canal 2

23760 3E 02           ld a,2

23762 CD 01 16       call 5633

ESTABLECER OVER 1

L1

23765 FD 36 57 03 ld (iy+87),3

CURSOR PARPADEA EN LAZO DE ESPERA

23769 3E 8F           ld a,143

23771 D7             , rst 16

23772 3E 08           ld a,8

23774 D7             rst 16

23775 3E 8F           ld a,143

23777 D7             rst 16

23778 3E 08           ld a,8

23780 D7             rst 16

ESPERAR PULSACION DE TECLA

23781 FD CB 01 6E bit 5, (iy+1)

23785 28 EA           jr z,L1

23787 FD CB 01 AE     res 5,(iy+1)

ESTABLECER OVER 0

23791 FD 36 57 00 ld (iy+87),0

OBTENER LAST KEY (ULTIMA TECLA)

23795 3A 08 5C       ld a,(23560)

EXAMINAR SI ES 'STOP'

23798 FE E2           cp 226

23800 C8             ret z

IMPRIMIR CHR\$

23801 D7             rst 16

23802 18 D9          jr L1

## Programa 5.5

# 6 MOVIMIENTO

Ahora que somos capaces de escribir en la pantalla y de leer el teclado podemos buscar métodos de animación. Esto es, por supuesto, parte esencial de cualquier programa de juegos en “tiempo real”. En este Capítulo estudiaremos métodos para dar la impresión de movimiento:

1. Movimiento de primer plano —barcos, marcianos, gente, etc.
2. Movimiento del fondo —estrellas, planetas, fragmentos de cielo, etc.
3. Movimiento de ambos, 1 y 2.

El tipo de movimiento empleado por la mayoría de las personas que hacen software es de “cuadro de carácter”; éste permite que los programas sean rápidos y la utilización de todos los colores. La pantalla se divide en 24 filas y 32 columnas, y, cuando se mueve un carácter, cambia desde su posición actual hasta la fila/columna adyacente; este movimiento es tosco pero no a sacudidas. Si se emplea el movimiento de pixel entonces, especialmente cuando se trata de movimientos de primer término, los caracteres deben redibujarse. Si ha utilizado la orden PLOT del BASIC se habrá dado cuenta de que normalmente nos quedamos restringidos al uso de dos colores (PAPER e INK).

## **Movimiento de primer plano de un cuadro de carácter**

El primer paso es definir (o inicializar) la posición de comienzo del carácter y, utilizando cualquiera de los métodos citados en el Capítulo 5, leer el teclado, y actualizar la posición del carácter de acuerdo con ello. La actualización de la posición del carácter es realizada en un “grupo duplicado” de datos de posiciones de comienzo. Cuando la actualización ha sido completada habrá dos grupos de datos: el primero contendrá la posición de pantalla actual y el segundo la nueva posición. Se llevan a cabo algunas pruebas, por ejemplo, para ver si la nueva posición está aún en la pantalla, y se realiza la corrección. Para conseguir la

impresión de movimiento de parpadeo-libre, sacamos el carácter existente y lo volvemos a visualizar *inmediatamente* en su nueva posición. El método que yo uso generalmente consiste en utilizar OVER 1 y REPRINT para volver a visualizar el carácter en la posición *anterior*; de esta forma, sacándolo, y utilizando todavía OVER 1 volvemos a repintar en la *nueva* posición. Esto nos permitirá mover el carácter sobre el fondo sin borrarlo.

org 23760		23835 01 0A 00	ld bc,10
DESHABILITAR TECLADO		23838 CD 3C 20	call 8252
		L7	
23760 F3	di	TRANSFERIR DATA 2 A STORE	
23761 18 18	jr START		
DATOS UDG			
DATA 1		23841 11 00 5B	ld de,23296
CHR\$ 144		23844 21 F9 5C	ld hl,DATA2
defb 0 0 15 24 49 226 225 224		23847 01 0A 00	ld bc,10
CHR\$ 145		23850 ED B0	
defb 24 60 255 255 153 255 231		ldir	
defb 60		OBTENER x,y DE STORE	
CHR\$ 146			
defb 0 0 240 24 140 71 135 7		23852 2A 01 5B	ld hl,(23297)
EMPEZAR		EXAMINAR SI SE PULSA TECLA	
ESTABLECER UDG's			
23787 ED 5B 7B 5C	ld de,(23675)	23855 01 FE EF	ld bc,61438
23791 21 D3 5C	ld hl,DATA1	23858 ED 78	in a,(c)
23794 01 18 00	ld bc,24	23860 CB 47	bit 0,a
23797 ED B0	ldir	23862 20 01	jr nz,L1
23799 18 0A	jr BEGIN	23864 24	inc h
Print CHR\$ DATA		L1	
AT x,y INK 8 PAPER 8		23865 CB 4F	bit 1,a
CHR\$ 144 145 146		23867 20 01	jr nz,L2
		23869 25	dec h
DATA2		L2	
defb 22 0 0 16 8 17 8		23870 01 FE F7	ld bc,63486
defb 144 145 146		23873 ED 78	in a,(c)
BEGIN		23875 CB 47	bit 0,a
BORDER 5		23877 20 01	jr nz,L3
23811 FD 36 0E 28 ld (iy+14),40		23879 2D	dec l
PAPER 1 INK 7		L3	
23815 FD 36 53 0F ld (iy+83),15		23880 01 FE FB	ld bc,64510
CLS			
23819 3E 02	ld a,2	23883 ED 78	in a,(c)
		23885 CB 47	bit 0,a
		23887 20 01	jr nz,L4
		23889 2C	inc 1
		L4	
23821 CD 01 16	call 5633	EXAMINAR SI ESTA QUIETO EN PANTALLA	
23824 CD 6B 0D	call 3435		
IMPRIMIR SERIE		23890 7D	ld a,l
		23891 FE 16	cp 22
23827 3E 02	ld a,2	23893 20 03	jr nz,L5
23829 CD 01 16	call 5633	23895 2D	dec 1
23832 11 F9 5C	ld de,DATA2	23896 18 05	jr L6
		L5	

23898 FE FF cp 255  
 23900 20 01 jr nz,L6  
 23902 2C inc l  
 L6  
 23903 7C ld a,h  
 23904 FE 1E cp 30  
 23906 20 03 jr nz,L8

23908 25 dec h  
 23909 18 05 jr L9  
 L8  
 23911 FE FF cp 255  
 23913 20 01 jr nz,L9  
 23915 24 inc h  
 L9

#### EXAMINAR SI NO SE MUEVE

23916 ED 5B 01 5B ld de,(23297)  
 23920 AF xor a  
 23921 E5 push hl  
 23922 ED 52 sbc hl,de  
 23924 E1 pop hl  
 SI NO SE MUEVE ENTONCES PASAR  
 POR ALTO PRINK OVER 1  
 23925 2B 2C jr z,L12

#### GUARDAR NUEVOS PARAMETROS x,y

23927 22 01 5B ld (23297),hl

#### ESTABLECER OVER 1

23930 FD 36 57 03 ld (iy+87),3

#### BORRAR POSICION ANTIGUA

23934 11 F9 5C ld de,DATA2  
 23937 01 0A 00 ld bc,10  
 23940 CD 3C 20 call 8252

#### REIMPRIMIR EN NUEVA POSICION

23943 11 00 5B ld de,23296  
 23946 01 0A 00 ld bc,10  
 23949 CD 3C 20 call 8252

#### ACTUALIZAR DATA2 PARA SIGUIENTE VEZ

23952 11 F9 5C ld de,DATA2

23955 21 00 5B ld hl,23296  
 23958 01 0A 00 ld bc,10  
 23961 ED B0 ldir

#### LAZO DE RETARDO

23963 21 88 13 ld hl,5000  
 L11  
 23966 2B dec hl  
 23967 7C ld a,h  
 23968 B5 or 1  
 23969 20 FB jr nz,L11

#### EXAMINAR SI PULSA 'SPACE'

L12  
 23971 01 FE 7F ld bc,32766  
 23974 ED 78 in a,(c)  
 23976 CB 47 bit 0,a  
 23978 C2 21 5D jp nz,L7

#### REINICIAR OVER 0

23981 FD 36 57 00 ld (iy+87),0

#### ACTIVAR TECLADO

23985 FB ei  
 23986 C9 ret

## Programa 6.1A

El programa 6.1A muestra el método explicado anteriormente. Se mueve un carácter, parecido a una nave espacial, sobre la pantalla, utilizando las teclas 9/0 para izquierda/derecha y 1/Q para arriba/abajo. Con la "barra de espaciado" volvemos al BASIC. La nave espacial puede moverse en dos direcciones a la vez y puede volar por la pantalla sin cambiar el fondo que anteriormente usted haya establecido. El carácter se imprime con INK 8; PAPER 8; y así tomará los colores de los cuadros según se defina el atributo de cada cuadro. La velocidad de movimiento es controlada con el lazo de retardo y se hace una prueba para ver si la nave está estacionaria. Si lo está, entonces se omite la rutina de revisualización ya que podría originar el parpadeo de la nave por el borrado y la revisualización continúa en la misma posición. Puede modificar el programa

para hacer posible que la nave vuele fuera de la pantalla y reaparezca por el otro lado, poniendo a cero "y" cuando la nave sobrepase los parámetros de la pantalla.

El movimiento de carácter múltiple es básicamente lo mismo que el anterior y cada carácter tiene su propio grupo de PRINT AT DATA. La rutina en código máquina es muy rápida, como puede demostrarse utilizando LD HL, 1 para el lazo de retardo, porque la TV no puede imprimir todas las posiciones cuando el carácter se mueve. Encontrará que la mayoría de los programas necesitan algún tipo de lazo de retardo para mantener la imagen durante un momentos antes de que cambie.

## **Movimiento de fondo de un cuadro de carácter**

Siempre que el movimiento de primer plano requiera guardar en memoria la posición del carácter y que siga la pista de los movimientos, el movimiento de fondo requerirá normalmente el desplazamiento de la visualización completa de la pantalla (o bloques de ella) en una dirección conocida sin tener que saber exactamente donde está cada cosa.

Este movimiento se realiza usualmente en una de las cuatro direcciones —arriba, abajo, izquierda y derecha— y requiere normalmente ambas cosas, los caracteres y sus atributos. Si un fondo de pantalla es desplazado por una posición de carácter tendremos una posterior elección concerniente a los caracteres desplazados fuera de la pantalla. Podemos eliminarlos e imprimir espacios en la fila/columna formada al otro lado de la pantalla, o podemos almacenarlos con sus atributos y reimprimirlos en su lugar correspondiente al otro lado de la pantalla, produciéndose así un efecto de enrosque, que es útil para los programas de juegos.

Los siguientes programas del 6.1B al 6.16 muestran un ROLL y un SCROLL en las cuatro direcciones; en primer lugar los atributos y después los caracteres de pantalla. Estos programas se pueden modificar fácilmente para efectuar un ROLL y un SCROLL para posiciones de pantalla, quedando el resto inmóvil. Observe la capa peculiar de la visualización de la pantalla que se divide en tres bloques de ocho filas. Cada fila se divide a su vez en ocho líneas de 32 bytes por línea. El mapa de memoria empieza en la fila 1 línea 1 y continúa con los 32 bytes de esa línea; después cambia a la fila 2 línea 1, fila 3 línea 1 y así sucesivamente hasta la fila 8 línea 1 completando el mapa de 256 bytes. Entonces se vuelve a saltar a la fila 1 línea 2, fila 2

línea 2, fila 3 línea 2, etc., y se termina con la fila 8 línea 8 llenando el mapa de 2K bytes.

Después cambiamos a las ocho filas del medio que son tratadas de la misma forma, y finalmente a las ocho filas últimas que también se direccionan así. Así se produce un “roll” y “scroll” de carácter un poco complicado y requiere el movimiento de 6K bytes.

org 23760		23773 3A 8D 5C	ld a,(23693)
23760 11 00 57	ld de,22528	L2	
23763 21 20 58	ld hl,22560	23776 12	ld (de),a
23766 01 E0 02	ld bc,736	23777 13	inc de
23769 ED B0	ldir	23778 10 FC	djnz L2
23771 06 20	ld b,32	23780 C9	ret

### Programa 6.1B

org 23760		23773 3A 8D 5C	ld a,(23693)
23760 11 FF 5A	ld de,23295	L7	
23763 21 DF 5A	ld hl,23263	23776 12	ld (de),a
23766 01 E0 02	ld bc,736	23777 13	inc de
23769 ED B8	ldir	23778 10 FC	djnz L7
23771 06 20	ld b,32	23780 C9	ret

### Programa 6.2

org 23760		23769 01 1F 00	ld bc,31
23760 06 18	ld b,24	23772 ED B0	ldir
23762 21 00 58	ld hl,22528	23774 3A 8D 5C	ld a,(23693)
L12		23777 12	ld (de),a
23765 E5	push hl	23778 C1	pop bc
23766 D1	pop de	23779 10 F0	djnz L12
23767 C5	push bc	23781 C9	ret
23768 23	inc hl		

### Programa 6.3

org 23760		23769 01 1F 00	ld bc,31
23760 06 18	ld b,24	23772 ED B8	ldir
23762 21 FF 5A	ld hl,23295	23774 3A 8D 5C	ld a,(23693)
L14		23777 12	ld (de),a
23765 E5	push hl	23778 C1	pop bc
23766 D1	pop de	23779 10 F0	djnz L14
23767 C5	push bc	23781 C9	ret
23768 2B	dec hl		

### Programa 6.4

org 23760		23772 01 E0 02	ld bc,736
23760 21 00 58	ld hl,22528	23775 ED B0	ldir
23763 E5	push hl	23777 06 20	ld b,32
23764 D1	pop de	L17	
23765 06 20	ld b,32	23779 F1	pop af
L16		23780 2B	dec hl
23767 7E	ld a,(hl)	23781 77	ld (hl),a
23768 F5	push af	23782 10 FB	djnz L17
23769 23	inc hl	23784 C9	ret
23770 10 FB	djnz L16		

### Programa 6.5

```
org 23760
23760 21 FF 5A
23763 E5
23764 D1
23765 06 20
L24
23767 7E
23768 F5
23769 2B
23770 10 FB
```

```
ld hl,23295
push hl
pop de
ld b,32

ld a,(hl)
push af
dec hl
djnz L24
```

```
23772 01 E0 02
23775 ED B8
23777 06 20
L25
23779 F1
23780 23
23781 77
23782 10 FB
23784 C9
```

```
ld bc,736
lddr
ld b,32

pop af
inc hl
ld (hl),a
djnz L25
ret
```

## Programa 6.6

```
org 23760
23760 06 18
23762 21 00 58
L32
23765 E5
23766 D1
23767 C5
23768 7E
```

```
ld b,24
lh hl,22528

push hl
pop de
push bc
ld a,(hl)
```

```
23769 23
23770 01 1F 00
23773 ED B0
23775 12
23776 C1
23777 10 F2
23779 C9
```

```
inc hl
ld bc,31
ldir
ld (de),a
pop bc
djnz L32
ret
```

## Programa 6.7

```
org 23760
23760 06 18
23762 21 FF 5A
L33
23765 E5
23766 D1
23767 C5
23768 7E
```

```
ld b,24
ld hl,23295

push hl
pop de
push bc
ld a,(hl)
```

```
23769 2B
23770 01 1F 00
23773 ED B8
23775 12
23776 C1
23777 10 F2
23779 C9
```

```
dec hl
ld bc,31
lddr
ld (de), a
pop bc
djnz L33
ret
```

## Programa 6.8

```
org 23760
23760 06 C0
23762 21 00 40
L1
23765 E5
23766 D1
23767 C5
23768 7E
```

```
ld b,192
ld hl,16384

push hl
pop de
push bc
ld a,(hl)
```

```
23769 23
23770 01 1F 00
23773 ED B0
23775 12
23776 C1
23777 10 F2
23779 C9
```

```
inc hl
ld bc,31
ldir
ld (de), a
pop bc
djnz L1
ret
```

## Programa 6.9

```
org 23760
23760 06 C0
23762 21 FF 57
L1
23765 E5
23766 D1
23767 C5
23768 7E
```

```
ld b,192
ld hl,22527

push hl
pop de
push bc
ld a,(hl)
```

```
23769 2B
23770 01 1F 00
23773 ED B8
23775 12
23776 C1
23777 10 F2
23779 C9
```

```
dec hl
ld bc,31
lddr
ld (de), a
pop bc
djnz L1
ret
```

## Programa 6.10



```

org 23760
23760 21 00 40      ld hl,16384
23763 AF            xor a
23764 06 C0          ld b,192
L13
23766 C5            push bc
23767 E5            push hl
23768 D1            pop de

```

```

23769 23
23770 01 1F 00      23773 ED B0
23775 12
23776 C1
23777 10 F3
23779 C9

```

```

inc hl
ld, bc,31
ldir
ld (de), a
pop bc
djnz L13
ret

```

## Programa 6.11

```

org 23760
L2
23760 21 00 40      ld hl,16384
23763 21 20 00      ld de,32
23766 06 C0          ld b,192
L1
23768 36 00          ld (hl),0
23770 19            add hl, de
23771 10 FB          djnz L1

```

```

23773 21 01 40      23776 11 00 40
23779 01 FF 17      23782 ED 80
23784 EB
23785 36 00
23787 C9

```

```

ld hl,16385
ld de,16384
ld bc,6143
ldir
ex de, hl
ld (hl),0
ret

```

## Programa 6.11A

```

org 23760
23760 21 FF 57      ld hl,22527
23763 AF            xor a
23764 06 C0          ld b,192
L14
23766 C5            push bc
23767 E5            push hl
23768 D1            pop de

```

```

23769 2B
23770 01 1F 00      23773 ED B8
23775 12
23776 C1
23777 10 F3
23779 C9

```

```

dec hl
ld bc,31
ldir
ld (de), a
pop bc
djnz L14
ret

```

## Programa 6.12

```

org 23760
23760 11 00 40      ld de,16384
23763 21 20 40      ld hl,16416
23766 01 E0 07      ld bc,2016
23769 ED B0          ldir
23771 06 08          ld b,8
23773 21 00 48      ld hl,18432
23776 11 E0 40      ld de,16608
L3
23779 C5            push bc
23780 E5            push hl
23781 D5            push de
23782 01 20 00      ld bc,32
23785 ED B0          ldir
23787 D1            pop de
23788 E1            pop hl
23789 C1            pop bc
23790 14            inc d
23791 24            inc h
23792 10 F1          djnz L3

```

```

23814 E5
23815 D5
23816 01 20 00      23819 ED B0
23821 D1
23822 E1
23823 C1
23824 14
23825 24
23826 10 F1
23828 11 00 50      23831 21 20 50

```

```

push hl
push de
ld bc,32
ldir
pop de
pop hl
pop bc
inc d
inc h
djnz L4
ld de,20480
ld hl,20512

```

```

23794 11 00 48      ld de,18432
23797 21 20 48      ld hl,18464
23800 01 E0 07      ld bc,2016
23803 ED B0          ldir
23805 06 08          ld b,8
23807 21 00 50      ld hl,20480
23810 11 E0 48      ld de,18656
L4
23813 C5            push bc

```

```

23834 01 E0 07      23837 ED B0
23839 06 08          23841 21 E0 50
L5
23844 C5
23845 E5
23846 06 20
L6
23848 36 00
23850 23
23851 10 FB
23853 E1
23854 C1
23855 24
23856 10 F2
23858 C9

```

```

ld bc,2016
ldir
ld b,8
ld hl,20704
push bc
push hl
ld b,32
ld (hl),0
inc hl
djnz L6
pop hl
pop bc
inc h
djnz L5
ret

```

## Programa 6.13

org 23760		23818 01 E0 07	ld bc,2016
23760 21 00 40	ld hl,16384	23821 ED B0	ldir
L18		23823 06 08	ld b,8
23763 06 20	ld b,32	23825 21 00 50	ld hl,20480
L19		23828 11 E0 48	ld de,18656
23765 7E	ld a,(hl)		
23766 F5	push af	L21	
23767 23	inc hl	23831 C5	push bc
23768 10 FB	djnz L19	23832 E5	push hl
23770 AF	xor a	23833 D5	push de
23771 6F	ld l,a	23834 01 20 00	ld bc,32
23772 24	inc h	23837 ED B0	ldir
23773 7C	ld a,h	23839 D1	pop de
23774 FE 48	cp 72	23840 E1	pop hl
23776 20 F1	jr nz,L18	23841 C1	pop bc
23778 11 00 40	ld de,16384	23842 14	inc d
23781 21 20 40	ld hl,16416	23843 24	inc h
23784 01 E0 07	ld bc,2016	23844 10 F1	djnz L21
23787 ED B0	ldir	23846 11 00 50	ld de,20480
23789 06 08	ld b,8	23849 21 20 50	ld hl,20512
		23852 01 E0 07	ld bc,2016
23791 21 00 48	ld hl,18432	23855 ED B0	ldir
23794 11 E0 40	ld de,16608	23857 21 FF 57	ld hl,22527
L20		L22	
23797 C5	push bc	23860 06 20	ld b,32
23798 E5	push hl	L23	
23799 D5	push de	23862 F1	pop af
23800 01 20 00	ld bc,32		
23803 ED B0	ldir	23863 77	ld (hl),a
		23864 2B	dec hl
23805 D1	pop de	23865 10 FB	djnz L23
23806 E1	pop hl	23867 3E FF	ld a,255
23807 C1	pop bc	23869 6F	ld l,a
23808 14	inc d	23870 25	dec h
23809 24	inc h	23871 7C	ld a,h
23810 10 F1	djnz L20	23872 FE 4F	cp 79
23812 11 00 48	ld de,18432	23874 20 F0	jr nz,L22
23815 21 20 48	ld hl,18464	23876 C9	ret

## Programa 6.14

org 23760		23805 06 08	ld b,8
23760 11 FF 57	ld de,22527	23807 21 E0 40	ld hl,16608
23763 21 DF 57	ld hl,22495	23810 11 00 48	ld de,18432
23766 01 E0 07	ld bc,2016	L9	
23769 ED B8	lddr	23813 C5	push bc
23771 06 08	ld b,8	23814 E5	push hl
23773 21 E0 48	ld hl,18656	23815 D5	push de
23776 11 00 50	ld de,20480	23816 01 20 00	ld bc,32
L8		23819 ED B0	ldir
23779 C5	push bc	23821 D1	pop de
23780 E5	push hl	23822 E1	pop hl
23781 D5	push de	23823 C1	pop bc
23782 01 20 00	ld bc,32	23824 14	inc d
23785 ED B0	ldir	23825 24	inc h
23787 D1	pop de	23826 10 F1	djnz L9
23788 E1	pop hl	23828 11 FF 47	ld de,18431
23789 C1	pop bc	23831 21 DF 47	ld hl,18399
23790 14	inc d		
23791 24	inc h	23834 01 E0 07	ld bc,2016
23792 10 F1	djnz L8	23837 ED B8	lddr
		23839 21 00 40	ld hl,16384
23794 11 FF 4F	ld de,20479	L10	
23797 21 DF 4F	ld hl,20447	23842 06 20	ld b,32
23800 01 E0 07	ld bc,2016	L11	
23803 ED B8	lddr	23844 36 00	ld (hl),0

23846 23 inc hl  
23847 10 FB djnz L11  
23849 AF xor a  
23850 6F ld l,a  
23851 24 inc h

23852 7C  
23853 FE 48  
23855 20 F1  
23857 C9

ld a,h  
cp 72  
jr nz,L10  
ret

## Programa 6.15

org 23760  
23760 21 FF 57 ld hl,22527  
L26  
23763 06 20 ld b,32  
L27  
23765 7E ld a,(hl)  
23766 F5 push af  
23767 2B dec hl  
23768 10 FB djnz L27  
23770 3E FF ld a,255  
23772 6F ld l,a  
23773 25 dec h  
23774 7C ld a,h  
23775 FE 4F cp 79  
23777 20 F0 jr nz,L26  
23779 11 FF 57 ld de,22527  
23782 21 DF 57 ld hl,22495  
23785 01 E0 07 ld bc,2016  
23788 ED B8 lddr  
23790 06 08 ld b,8  
23792 21 E0 48 ld hl,18656  
23795 11 00 50 ld de,20480  
L28  
23798 C5 push bc  
23799 E5 push hl  
23800 D5 push de  
23801 01 20 00 ld bc,32  
23804 ED B0 ldir  
23806 D1 pop de  
23807 E1 pop hl  
23808 C1 pop bc  
23809 14 inc d  
23810 24 inc h  
23811 10 F1 djnz L28  
23813 11 FF 4F ld de,20479  
23816 21 DF 4F ld hl,20447  
23819 01 E0 07 ld bc,2016  
23822 ED B8 lddr

23824 06 08  
23826 21 E0 40  
23829 11 00 48

ld b,8  
ld hl,16608  
ld de,18432

L29  
23832 C5  
23833 E5  
23834 D5  
23835 01 20 00  
23838 ED B0  
23840 D1  
23841 E1  
23842 C1  
23843 14  
23844 24  
23845 10 F1  
23847 11 FF 47  
23850 21 DF 47  
23853 01 E0 07  
23856 ED B8  
23858 21 00 40  
L30  
23861 06 20  
L31  
23863 F1

push bc  
push hl  
push de  
ld bc,32  
ldir  
pop de  
pop hl  
pop bc  
inc d  
inc h  
djnz L29  
ld de, 18431  
ld hl, 18399  
ld bc, 2016  
lddr  
ld hl,16384

ld b,32  
pop af

23864 77  
23865 23  
23866 10 FB  
23868 AF  
23869 6F  
23870 24  
23871 7C  
23872 FE 48  
23874 20 F1  
23876 C9

ld (hl),a  
inc hl  
djnz L31  
xor a  
ld l,a  
inc h  
ld a,h  
cp 72  
jn nz L30  
ret

## Programa 6.16

1 REM >?PASO £? BORRAR 65/>?  
STEP ?  
500>DATA 24,79,5,2,"PARAR LA CINTA"  
510 DATA 0,167,6,4, "MOVIMIENTO"  
520 DATA 0,130,2,4, "-----"  
525 DATA 24,15,1,2, "PULSE CUALQUIER  
TECLA"  
550 DATA 31,170,1,2,"INSTRUCCIONES"  
600 BORDER 4: PAPER 6: INK 0: CLS  
605 RESTORE 500  
610 FOR a=1 TO 4  
620 INK a: GO SUB 8000  
630 PAUSE 50. NEXT a  
640 PAUSE 0: CLS  
642 INK 2: GO SUB 8000  
645 PRINT INK 0; AT 3,0; "Este programa  
muestra los métodos básicos de movimientos  
de primer plano y de fondo."

```

646 PRINT "La tabla de abajo muestra las
teclas usadas y sus funciones."
647 PRINT
650 PRINT "    FONDO"
655 PRINT "    ROLL ; SCROLL
NAVE ESPACIAL"
660 PRINT "ARRIBA    1    :    Q
7"
670 PRINT "ABAJO    2    :    W
8"
680 PRINT "IZQUIERDA    3    :    E
9"
690 PRINT "DERECHA 4:R
0"
695 PRINT "Con la tecla SPACE se vuelve
al BASIC"
700 RESTORE 525: INK 2: GO SUB 8000
705 IF INKEY$<>"" THEN GO TO 705
710 IF INKEY$="" THEN GO TO 710
715 POKE 30046,0: POKE: 30047,0
720 RANDOMIZE USR 30001
725 CLS : RESTORE 730: PAPER 8:
INK 0: GO SUB 8000
730 DATA 0,80,3,1 "ENTER 'GOTO 7
15' para otra vez".
740 STOP
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 30000: LOAD "udg" CODE
USR "a": LOAD ""CODE : GO TO 500
9999 SAVE "MOVEDEMO" LINE 9998: S
AVE "udg" CODE USR "a", 168: SAVE "
M/code "CODE 30001,2767

```

## Programa 6.17

### INSTRUCCIONES

Este programa muestra los métodos básicos de movimiento de primer plano y de fondo.

La tabla de abajo muestra las teclas usadas y sus funciones.

	FONDO		
	ROLL	SCROLL	NAVE ESPACIAL
ARRIBA	1	Q	7
ABAJO	2	W	8
IZQUIERDA	3	E	9
DERECHA	4	R	0

con la Tecla SPACE se vuelve al BASIC

PULSE CUALQUIER TECLA

org 40000 30001		LP7	
30001 F3	di	30128 FE FF	cp 255
30002 CD D0 5C	call 23760	30130 20 01	jr nz,LP8
30005 18 18	jr COMIENZO	30132 24	inc h
DATA 1		LP8	
defb 0 0 15 24 49 226 225 224		30133 ED 5B 01 5B	ld de,(23297)
defb 24 60 255 255 153 255 231		30137 AF	xor a
defb 60 0 0 240 24 14071 135 7		30138 E5	push hl
COMIENZO		30139 ED 52	sbc hl,de
30031 ED 5B 7B 5C	ld de,(23675)	30141 E1	pop hl
30035 21 37 75	ld hl,DATA 1	30142 28 24	jr z,LP12
30038 01 18 00	ld bc,24	30144 22 01 5B	ld (23297),hl
30041 ED B0	ldir	OVER1	
30043 18 0A	jr COMENZAR	30147 FD 36 57 03	ld (iy+87),3
DATA2		30151 11 5D 75	ld de,DATA2
defb 22 0 0 16 8 17 8 144 145		30154 01 0A 00	ld bc,10
defb 146		30157 CD 3C 20	call 8252
COMENZAR		30160 11 00 5B	ld de,23296
30055 11 5D 75	ld de,DATA2	30163 01 0A 00	ld bc,10
30058 01 0A 00	ld bc,10	30166 CD 3C 20	call 8252
		30169 11 5D 75	ld, de DATA2
30061 CD 3C 20	call 8252		
30064 11 00 5B	ld de,23296		
30067 21 5D 75	ld hl,DATA2	3072 21 00 5B	ld hl, 23296
30070 01 0A 00	ld bc,10	30175 01 0A 00	ld bc,10
30073 ED B0	ldir	30178 ED 80	ldir
MOVER		LP12	
30075 2A 01 5B	ld hl,(23297)	30180 01 FE F7	ld bc,63486
30078 01 FE EF	ld bc,61438	30183 ED 78	in a,(c)
30081 ED 78	in a,(c)	30185 E6 0F	and 15
30083 FE FF	cp 255	30187 28 3E	jr z,SCR
30085 28 5D	jr z,LP12	30189 FE 03	cp 3
30087 CB 47	bit 0,a	30191 28 3A	jr z,SCR
		30193 FE 0C	cp 12
30089 20 01	jr nz,LP1	30195 28 36	jr z,SCR
30091 24	inc h	30197 FE 0F	cp 15
LP1		30199 28 32	jr z,SCR
30092 CB 4F	bit 1,a	30201 F5	push af
30094 20 01	jr nz,LP2	30202 FD 36 57 03	ld (iy+87),3
30096 25	dec h	30206 11 5D 75	ld de,DATA2
LP2		30209 01 0A 00	ld bc,10
30097 CB 57	bit 2,a	30212 CD 3C 20	call 8252
30099 20 01	jr nz,LP3	30215 F1	pop af
		30216 CB 47	bit 0,a
30101 2C	inc l		
LP3		30218 F5	push af
30102 CB 5F	bit 3,a	30219 CC C9 77	call z,UROLL
30104 20 01	jr nz,LP4	30222 F1	pop af
30106 2D	dec l	30223 CB 4F	bit 1,a
LP4		30225 F5	push af
30107 7D	ld a,1	30226 CC 56 78	call z, DROLL
30108 FE 16	cp 22	30229 F1	pop af
30110 20 03	jr nz,LP5	30230 CB 57	bit 2,a
30112 2D	dec l	30232 F5	push af
30113 18 05	jr LP6	30233 CC E3 78	call z,LROLL
LP5		30236 F1	pop af
30115 FE FF	cp 255	30237 CB 5F	bit 3,a
30117 20 01	jr nz,LP6	30239 CC F7 78	call z,RROLL
30119 2C	inc l	30242 11 5D 75	ld de,DATA 2
LP6		30245 01 0A 00	ld bc,10
30120 7C	ld a,h	30248 CD 3C 20	call 8252
30121 FE 1E	cp 30	SCR	
30123 20 03	jr nz,LP7	30251 01 FE FB	ld bc,64510
30125 25	dec h	30254 ED 78	in a,(c)
30126 18 05	jr LP8	30256 E6 0F	and 15
		30258 28 3E	jr z,DELAY

30260 FE 03	cp 3	30385 C5	push bc
30262 28 3A	jr z,DELAY		
30264 FE 0C	cp 12	30386 E5	push hl
30266 28 36	jr z,DELAY	30387 D5	push de
30268 FE 0F	cp 15	30388 01 20 00	ld bc,32
30270 28 32	jr z,DELAY	30391 ED B0	ldir
30272 F5	push af	30393 D1	pop de
30273 FD 36 57 03	ld (iy+87),3	30394 E1	pop hl
30277 11 5D 75	ld de,DATA2	30395 C1	pop bc
30280 01 0A 00	ld bc,10	30396 14	inc d
30283 CD 3C 20	call 8252	30397 24	inc h
30286 F1	pop af	30398 10 F1	djnz L3
30287 CB 47	bit 0,a	30400 11 00 48	ld de,18432
30289 F5	push af	30403 21 20 48	ld hl,18464
30290 CC 8A 76	call z, USCR	30406 01 E0 07	ld bc,2016
30293 F1	pop af	30409 ED B0	ldir
30294 CB 4F	bit 1,a	30411 06 08	ld b,8
30296 F5	push af	30413 21 00 50	ld hl,20480
30297 CC 01 77	call z,DSCR	30426 11 E0 4B	ld de,18656
30300 F1	pop af	L4	
30301 CB 57	bit 2,a	30419 C5	push bc
		30420 E5	push hl
		30421 D5	push de
30303 F5	push af		
30304 CC 77 77	call z,LSCR	30422 01 20 00	ld bc,32
30307 F1	pop af	30425 ED B0	ldir
30308 CB 5F	bit 3,a	30427 D1	pop de
30310 CC A0 77	call z,RSCR	30428 E1	pop hl
30313 11 5D 75	ld de,DATA2	30429 C1	pop bc
30316 01 0A: 00	ld bc,10	30430 14	inc d
30319 CD 3C 20	call 8252	30431 24	inc h
DELAY		30432 10 F1	djnz L4
30322 21 10 27	ld hl,10000	30434 11 00 50	ld de,20480
L1		30437 21 20 50	ld hl,20512
30325 2B	dec hl	30440 01 E0 07	ld bc,2016
30236 7C	ld a,h	30443 ED B0	ldir
30327 B5	or l	30445 06 08	ld b,8
30328 20 FB	jr nz, L1	30447 21 E0 50	ld, hl,20704
30330 01 FE 7F	ld bc,32766	L5	
30333 ED 78	in a,(c)	30450 C5	
30335 CB 47	bit 0,a	30541 E5	push bc
30337 C2 7B 75	jp nz, MOVE	30452 06 20	push hl
30340 FD 36 57 00	ld (iy+87),0	L6	ld b,32
30344 FB	ei	30454 36 00	ld (hl),0
		30456 23	inc hl
30345 C9	ret		
USCR		30457 10 FB	djnz L6
30346 11 00 58	ld de,22528	30459 E1	pop hl
30349 21 20 58	ld hl,22560	30460 C1	pop bc
30352 01 E0 02	ld bc,736	30461 24	inc h
30355 ED B0	ldir	30462 10 F2	dinz L5
30357 06 20	ld b,32	30464 C9	ret
30359 3A 8D 5C	ld a,(23693)	DSCR	
L2		30465 11 FF 5A	ld de,23295
30362 12	ld (de),a	30468 21 DF 5A	ld hl,23263
30363 13	inc de	30471 01 E0 02	ld bc,736
30364 10 FC	djnz L2	30474 ED B8	lddr
30366 11 00 40	ld de,16384	30476 06 20	ld b,32
30369 21 20 40	ld hl,16416	30478 3A 8D 5C	ld a,(23693)
30372 01 E0 07	ld bc,2016	L7	
30375 ED 80	ldir	30481 12	ld (de),a
30377 06 08	ld b,8	30482 13	inc de
30379 21 00 48	ld hl,18432	30482 10 FC	djnz L7
30382 11 E0 40	ld de,16608	30485 11 FF 57	ld de, 22527
L3		30488 21 DF 57	ld hl,22495

30491 01 E0 07	ld bc,2016	30590 C5	push bc
30494 ED B8	lddr	30591 23	inc hl
		30592 01 1F 00	ld bc,31
		30595 ED B0	ldir
		30597 3A 8D 5C	ld a, (23693)
30496 06 08	ld b,8	30600 12	ld (de),a
30498 21 E0 48	ld hl,18656	30601 C1	pop bc
30501 11 00 50	ld de,20480	30602 10 F0	djnz L12
L8		30604 21 00 40	ld hl,16384
30504 C5	push bc	30607 AF	xor a
30505 E5	push hl	30608 06 C0	ld b,192
30506 D5	push de	L13	
30507 01 20 00	ld bc,32	30610 C5	push bc
30510 ED B0	ldir	30611 E5	push hl
30512 D1	pop de	30612 D1	pop de
30513 E1	pop hl	30613 23	inc hl
30514 C1	pop bc	30614 01 1F 00	ld bc,31
30515 14	inc d	30617 ED 80	ldir
30516 24	inc h	30619 12	ld (de),a
30517 10 F1	djnz L8	30620 C1	pop bc
30519 11 FF 4F	ld de,20479	30621 10 F3	djnz L13
30522 21 DF 4F	ld, hl,20447	30623 C9	ret
30525 01 E0 07	ld bc,2016	RSCR	
30528 ED B8	lddr	30624 06 18	ld b,24
30530 06 08	ld b,8	30626 21 FF 5A	ld hl,23295
30552 21 E0 40	ld hl,16608	L14	
		30629 E5	push hl
30535 11 00 48	ld de,18432	30630 D1	pop de
L9		30631 C5	push bc
30538 C5	push bc	30632 2B	dec hl
30539 E5	push hl	30633 01 1F 00	ld bc,31
30540 D5	push de	30636 ED B8	lddr
30541 01 20 00	ld bc,32	30638 3A 8D 5C	ld a,(23693)
30544 ED B0	ldir	30641 12	ld (de),a
30546 D1	pop de	30642 C1	pop bc
30547 E1	pop hl	30643 10 F0	djnz L14
30548 C1	pop bc	30645 21 FF 57	ld hl,22527
30549 14	inc d	30648 AF	xor a
30550 24	inc h	30649 06 C0	ld b,192
30551 10 F1	djnz L9	L14	
30553 11 FF 47	ld de,18431	30651 C5	push bc
30556 21 DF 47	ld hl,18399	30652 E5	push hl
30559 01 E0 07	ld bc,2016	30653 D1	pop de
30562 ED B8	lddr	30654 2B	dec hl
30564 21 00 40	ld hl,16384	30655 01 1F 00	ld bc,31
L10		30658 ED B8	lddr
30567 06 20	ld b,32	30660 12	ld (de),a
L11			
30569 36 00	ld (hl),0	30661 C1	pop bc
30571 23	inc hl	30662 10 DD	djnz L14
30572 10 FB	djnz L11	30664 C9	ret
30574 AF	xor a	UROLL	
30575 6F	ld l,a	30665 21 00 58	ld hl,22528
30576 24	inc h	30668 E5	push hl
30577 7C	ld a,h	30669 D1	pop de
30578 FE 48	cp 72	30670 06 20	ld b,32
30580 20 F1	jr nz,L10	L16	
30582 C9	ret	30672 7E	ld a,(hl)
LSCR		30673 F5	push af
30583 06 18	ld b,24	30674 23	inc hl
30585 21 00 58	ld hl,22528	30675 10 FB	djnz L16
L12		30677 01 E0 02	ld bc,736
30588 E5	push hl	30680 ED B0	ldir
30589 D1	pop de		

30682 06 20 L17	ld b,32	30781 01 E0 07	ld bc,2016
30684 F1	pop af	30784 ED B0	ldir
30685 2B	dec hl	30786 21 FF 57	ld hl,22527
30686 77	ld (hl),a	L22	
30687 10 FB	djnz L17	30789 06 20	ld b,32
		L23	
		30791 F1	pop af
30689 21 00 40	ld l,16384	30792 77	ld (hl),a
L18		30793 2B	dec hl
30692 06 20	ld b,32	30794 10 FB	djnz L23
L19		30796 3E FF	ld a,255
30694 7E	ld a,(hl)	30798 6F	ld l,a
30695 F5	push af	30799 25	dec h
30696 23	inc hl	30800 7C	ld a,h
30697 10 FB	djnz L19	30801 FE 4F	cp 79
30699 AF	xor a	30803 20 F0	jr nz,L22
30700 6F	ld, la	30805 C9	ret
30701 24	inc h	DROL	
30702 7C	ld a,h	30806 21 FF 5A	ld hl,23295
30703 FE 48	cp 72	30809 E5	push hl
30705 20 F1	jr nz,L18	30810 D1	pop de
30707 11 00 40	ld de,16384	30811 06 20	ld b,32
30710 21 20 40	ld hl,16416	L24	
30713 01 E0 07	ld bc,2016	30813 7E	ld a,(hl)
30716 ED B0	ldir	30814 F5	push af
30718 06 08	ld b,8	30815 2B	dec hl
30720 21 00 48	ld hl,18432	30816 10 FB	djnz L24
30723 11 E0 40	ld de,16608	30818 01 E0 02	ld bc,736
		30821 ED B8	lddr
		30823 06 20	ld b,32
L20			
30726 C5	push bc	L25	
30727 E5	push hl	30825 F1	pop af
30728 D5	push de	30826 23	inc hl
30729 01 20 00	ld bc,32	30827 77	ld (hl),a
30732 ED B0	ldir	30828 10 FB	djnz L25
30734 D1	pop de	30830 21 FF 57	ld hl,22527
30735 E1	pop hl	L26	
30736 C1	pop bc	30833 06 20	ld b,32
30737 14	inc d	L27	
30738 24	inc h	30835 7E	ld a,(hl)
30739 10 F1	djnz L20	30836 F5	push af
30741 11 00 48	ld de,18432	30837 2B	dec hl
30744 21 20 48	ld hl,18464	30838 10 FB	djnz L27
30747 01 E0 07	ld bc,2016	30840 3E FF	ld a,255
30750 ED B0	ldir	30842 6F	ld l,a
30752 06 08	ld b,8	30843 25	dec h
30754 21 00 50	ld hl,20480	30844 7C	ld a,h
30757 11 E0 48	ld de,18656	30845 FE 4F	cp 79
L21		30847 20 F0	jr nz,L26
30760 C5	push bc	30849 11 FF 57	ld de,22527
		30852 21 DF 57	ld hl, 22495
30761 E5	push hl		
30762 D5	push de	30855 01 E0 07	ld bc,2016
30763 01 20 00	ld b,32	30858 ED B8	lddr
30766 ED B0	ldir	30860 06 08	ld b,8
30768 D1	pop de	30862 21 E0 48	ld hl,18656
30769 E1	pop hl	30865 11 00 50	ld de,20480
30770 C1	pop bc	L28	
30771 14	inc d	30868 C5	push bc
30772 24	inc h	30869 E5	push hl
30773 10 F1	djnz L21	30870 D5	push de
30775 11 00 50	ld de,20480	30871 01 20 00	ld bc,32
30778 21 20 50	ld hl,20512	30874 ED B0	ldir



30876 D1	pop de	30936 10 FB	djnz L31
30877 E1	pop hl	30938 AF	xor a
30878 C1	pop bc	30939 6F	ld l,a
30879 14	inc d	30940 24	inch h
30880 24	inc h	30941 7C	ld a,h
30881 10 F1	djnz L28	30942 FE 48	cp 72
		30944 20 F1	jr nz,L30
		30946 C9	ret
30883 11 FF 4F	ld de,20479	LROLL	
30886 21 DF 4F	ld hl,20447	30947 06 D8	ld b,216
30889 01 E0 07	ld bc,2016	30949 21 00 40	ld hl,16384
30892 ED B8	lddr	L32	
		30952 E5	push hl
30894 06 08	ld b,8	30953 D1	pop de
30896 21 E0 40	ld hl,16608	30954 C5	push bc
30899 11 00 48	ld de,18432	30955 7E	ld a,(hl)
L29			
30902 C5	push bc	30956 23	inc hl
30903 E5	push hl	30957 01 1F 00	ld bc,31
30904 D5	push de	30960 ED B0	ldir
30905 01 20 00	ld bc,32	30962 12	ld (de),a
30908 ED B0	ldir	30963 C1	pop bc
30910 D1	pop de	30964 10 F2	djnz L32
30911 E1	pop hl	30966 C9	ret
30912 C1	pop bc	RROLL	
30913 14	inc d	30967 06 D8	ld b,216
30914 24	inc h	30969 21 FF 5A	ld hl,23295
30915 10 F1	djnz L29	L33	
30917 11 FF 47	ld de,18431	30972 E5	push hl
30920 21 DF 47	ld hl,18399	30973 D1	pop de
30923 01 E0 07	ld bc,2016	30974 C5	push bc
30926 ED B8	lddr	30975 7E	ld a,(hl)
30928 21 00 40	ld hl,16384	30976 2B	dec hl
L30		30977 01 1F 00	ld bc,31
		30980 ED B8	lddr
30931 06 20	ld b,32	30982 12	ld (de),a
L31		30983 C1	pop bc
30933 F1	pop af	30984 10 F	djnz L33
30934 77	ld (hl),a		
30935 23	inc hl	30986 C9	ret

## Programa 6.18

Observe que en estas rutinas se hace uso extensivo de las instrucciones LDIR y LDDR de transferencia de bloque, y que después de estas instrucciones los registros DE y HL contienen la dirección final + 1 (LDIR) o - 1 (LDDR). Se utiliza este hecho para volver a cargar HL o DE con una dirección específica para posteriores rutinas del programa. Por ejemplo, ATTR. UP SCROLL (programa 6.1 B) utiliza el hecho de que después de LDIR el registro DE contiene la dirección del atributo "START of LINE 24" y de esta forma no es necesario volver a cargar DE con esta dirección. Los atributos actuales de la línea 24 son reemplazados por los atributos permanentes de pantalla guardados en la variable ATTR P (23693) del sistema.

Para mostrar cada tipo de movimiento he escrito el programa

6.17 que viene en la cinta "MOVEDEMO". Como el programa tiene casi 1000 bytes de largo no es práctico guardarlo todo en una sentencia REM (ocuparía más de una pantalla completa), entonces lo he escrito para utilizarlo por encima de RAMTOP comenzando en la dirección 30001 (programa 6.18).

El código máquina en la sentencia REM se toma del programa de números aleatorios (programa 4.1) y es utilizado para establecer el fondo de pantalla. El programa es auto-explicativo y hace llamadas a diversas rutinas, dependiendo de las teclas que se pulsán. Los atributos cambian al mismo tiempo que los caracteres que se visualizan. Usted puede mover el fondo y la nave diagonalmente y si se pulsán dos teclas opuestas, entonces la rutina es pasada por alto. Pulsando "SPACE" se vuelve al BASIC. Observe que el movimiento vertical es más bien desigual y no es realmente apropiado debido a la transición de una zona de pantalla a otra. De hecho, los programas ponen la línea 1 en la línea 8 momentáneamente antes de desplazar la línea 9 hacia la línea 8.

Los programas de pantalla UP/DOWN-SCROLL y ROLL (programas de 6.13 a 6.16) fueron escritos de forma que puedan ser fáciles de seguir por usted sin subrutinas para hacerlos dependientes de la posición. Para conseguir un movimiento más suave estas rutinas deben abordarse de forma diferente, esto es, sacando una fila de una vez, en vez de sacar la línea superior de cada columna, luego la segunda línea de cada fila, etc., como se ha hecho hasta ahora. Para mostrar como se consigue esto he reescrito el programa SCREEN UP SCROLL (programa 6.19A). Como puede ver contiene una subrutina y una sub-subrutina, y es bastante más complicado. Quizás quiera usted reescribir la rutina SCREEN DOWN SCROLL de la misma forma, como ejercicio para estar seguro de que comprende completamente como funciona.

org 23760		L4	
23760 21 00 40	ld hl,16384	23793 E5	push hl
23763 CD 01 5D	call SECT	23794 06 20	ld b,32
23766 EB	ex de,hl	L3	
23767 21 00 48	ld hl,18432		
23770 E5	push hl		
23771 CD 15 5D	call UP		
23774 E1	pop hl	23796 36 00	ld (hl),0
23775 CD 01 5D	call SECT	23798 23	inc hl
23778 EB	ex de,hl	23799 10 FB	djnz L3
23779 21 00 50	ld hl,20480	23801 E1	pop hl
23782 E5	push hl	23802 24	inc h
23783 CD 15 5D	call UP	23803 7C	ld a,h
23786 E1	pop hl	23804 FE 58	cp 88
23787 CD 01 5D	call SECT		
23790 21 E0 50	ld hl,20704	23806 20 F1	jr nz,L4

23808 C9	ret	23828 C9	ret
SECT		UP	
23809 06 07	ld b,7	23829 06 08	ld b,8
L1		L2	
23811 C5	push bc	23831 C5	push bc
23812 01 20 00	ld bc,32	23832 E5	push hl
23815 E5	push hl	23833 D5	push de
23816 D1	pop de	23834 01 20 00	ld bc,32
23817 09	add hl,bc	23837 ED B0	ldir
23818 E5	push hl	23839 D1	pop de
23819 D5	push de	23840 E1	pop hl
23820 CD 15 5D	call UP		
23823 D1	pop de	23841 C1	pop bc
		23842 14	inc d
23824 E1	pop hl	23843 24	inc h
23825 C1	pop bc	23844 10 F1	djnz L2
23826 10 EF	djnz L1	23846 C9	ret

## Programa 6.19A

Si compara los programas 6.13 y 6.19A verá que hemos reducido la memoria requerida de 99 bytes a 87 bytes, pero aún no es tan compacto como podría serlo. Para mostrar cómo podríamos reducir la memoria y el tiempo de ejecución todavía más y obtener una característica útil extra consideraremos el programa 6.14, UP ROLL, que tal como está toma 117 bytes. Hasta ahora el programa maneja cada fila de carácter línea por línea y tiene el “fallo” de que tiene que transferir temporalmente la fila 1 a la fila 8 y la fila 9 a la fila 17. Podemos reescribirlo sobre las líneas del programa 6.21 para superar este problema, pero en vez de ello abordaremos el asunto con un enfoque completamente diferente para obtener un programa de sólo 76 bytes de largo.

En vez de manejar la pantalla con una fila cada vez, podemos escribir un programa que saque una columna cada vez. Entonces la rutina debe repetirse para las 32 columnas. De esta forma seremos capaces de seleccionar la columna que queramos listar, ya sea en bloques o una a una, y lo que es más, no requeriremos 256 bytes de memoria para almacenar la fila superior de la pantalla como en el programa 6.14, pues en vez de ello emplearemos sólo ocho bytes para el carácter de la columna superior. El programa 6.19B muestra como se realiza esta reducción de memoria.

org 23760  
CAPITULO 6  
PROGRAMA 19B

LA PANTALLA PRESENTA 'n' COLUMNAS  
EJEMPLO

PRESENTAR DESDE LAS COMUNNAS 5 A  
14 INC  
23760 AF xor a

LD DE,16383+COLUMNA DE COMIENZO  
1era COLUMNA = 1

23761 11 04 40	ld de,16388	TRANSFERIR ZONA DE PANTALLA	
LD B,Nu. de columnas		23797 21 20 07	ld hl,1824
23764 06 0A	ld b,10	23800 19	add hl,de
L5		23801 7C	ld a,h
23766 D5	push de	EXAMINAR SI ESTA FUERA DE PANTALLA	
23767 C5	push bc	por ejemplo COMIENZO DE ATRIBUTOS	
GUARDAR EL CARACTER DE LA LINEA SUPERIOR EN LA PILA		23802 FE 58	cp 88
23768 D5	push de	23804 28 0C	jr z,END
23769 E1	pop hl	23806 E5	push hl
23770 06 08	ld b,8	23807 06 08	ld b,8
L6		L3	
23772 7E	ld a,(hl)	23809 7E	ld a,(hl)
23773 F5	push af	23810 12	ld (de),a
23774 24	inc h	23811 24	inc h
23775 10 FB	djnz L6	23812 14	inc d
TRANSFERIR 7 LINEAS		23813 10 FA	djnz L3
L4		23815 D1	pop de
23777 06 07	ld b,7	23816 18 D7	jr L4
L2		SACAR CARACTER DE LA PILA Y PONERLO EN LA FILA INFERIOR DE LA PANTALLA	
23779 C5	push bc	FIN	
23780 21 20 00	ld hl,32	23818 11 20 00	ld de,32
23783 19	add hl,de	23821 ED 52	sbc hl,de
23784 E5	push hl	23823 06 08	ld b,8
TRANSFERIR 8 BYTES		L7	
23785 06 08	ld b,8	23825 F1	pop af
L1		23826 77	ld (hl),a
23787 7E	ld a,(hl)	23827 25	dec h
23788 12	ld (de),a	23828 10 FB	djnz L7
23789 24	inc h	MOVERSE A LA SIGUIENTE COLUMNA Y REPETIR	
23790 14	inc d	23830 C1	pop bc
23791 10 FA	djnz L1	23831 D1	pop de
23793 D1	pop de	23832 13	inc de
23794 C1	pop bc	23833 10 BB	djnz L5
23795 10 EE	djnz L2	23835 C9	ret

## Programa 6.19B

Observará que empezamos el programa 6.19B cargando el par de registro DE con la dirección del byte superior de la columna de comienzo, y el registro B con el número de columnas a listar. El programa es completamente auto-explicativo, y puede posicionarse en cualquier parte de la memoria.

Aún no hemos terminado de ahorrar memoria. ¿Podría creer que es posible reducir el programa 6.19B a 58 bytes y añadir una facilidad extra para poder definir las filas requeridas, y

conseguir una rutina que visualice una ventana de líneas? Pues bien, el programa 6.19C hace justamente eso.

Se ahorra memoria haciendo un examen, cuando se llega al final de una zona de pantalla, por medio de la instrucción BIT 0,H. Si BIT 0,H no es cero entonces tenemos que ir hasta el final de una zona, y en vez de sumar 32d a DE para mover hacia arriba una línea dentro de una zona, sumamos 1824 para movernos hasta la "primera" línea de la siguiente zona. Utilizando este método podemos definir cuantas filas requieren desplazarse, mediante el número de veces que se repite el lazo L4. Al principio se definen ROW/COLUMN con la dirección de visualización contenida en el par de registros DE (23671/3). Para calcular esta dirección de comienzo utilice la siguiente fórmula:

$$16384 + (1824 * \text{INT}(\text{ROW}/8)) = ((\text{ROW} - \text{INT}(\text{ROW}/8)) * 32) + \text{COLUMN}$$

En esta fórmula se emplean los números ROW/COLUMN de la Spectrum, es decir 1.<sup>a</sup> fila = 0, 1.<sup>a</sup> columna = 0.

**Ejemplo:** ROW 17 COLUMN 5  
 $= 16384 + (1824 * 2) + ((17 - 6) * 32) + 5$   
 $= 16384 + 3648 + 32 + 5$   
 $= 20069$

org 23760		23770 06 08	ld b,8
CAPITULO 6		L1	
PROGRAMA 19C		23772 7E	ld a,(hl)
		23773 F5	push af
VENTANA RODANTE HACIA ARRIBA		23774 24	inc h
		23775 10 FB	djnz L1
EJEMPLO			
COLUMNAS 4 A 10		LD B,N. de filas-1!	
FILAS 5 A 18			
		23777 06 0D	ld b,13
23760 AF	xor a	L4	
		23779 C5	push bc
LD DE,DIRECCION COMIENZO PANTALLA		23780 21 20 00	ld hl,32
por ejemplo DIRECCION COMIENZO		23783 19	add hl,de
COLUMNA/FILA			
		EXAMINAR CAMBIO DE ZONA	
23761 11 83 40	ld de,16515	DE PANTALLA	
LD B,N. de columnas			
		23784 CB 44	bit 0,h
23764 06 07	ld b,7	23786 28 04	jr z,L2
		23788 21 20 07	ld hl,1824
		23791 19	add hl,de
L6		L2	
23766 D5	push de	23792 E5	push hl
23767 C5	push bc	23793 06 08	ld b,8
23768 D5	push de	L3	
23769 E1	pop hl	23795 7E	ld a,(hl)

23796 12	ld (de),a	23808 F1	pop af
23797 14	inc d		
23798 24	inc h	23809 77	ld (hl),a
23799 10 FA	djnz L3	23810 10 FB	djnz L5
23801 D1	pop de	23812 C1	pop bc
23802 C1	pop bc	23813 D1	pop de
23803 10 F6	djnz L4	23814 D1	pop de
23805 06 08	ld b,8	23815 13	inc de
L5		23816 10 CC	djnz,L6
23807 25	dec h	23818 C9	ret

## Programa 6.19C

Para visualizar todas las líneas de la pantalla DE debe contener 16384, el registro B (antes de L6) debe contener 32 y el registro B (antes de L4) debe contener 23. Este programa se puede adaptar fácilmente para obtener las rutinas UP/DOWN SCROLL y DOWN ROLL.

## Movimiento de pixel

Este tipo de movimiento de caracteres de primer plano no es realmente aconsejable para programas de juegos, ya que suele ser más bien lento. El movimiento se reduce en un factor de ocho si, por ejemplo, trata de ver si un carácter golpea a un obstáculo; entonces cada bit de la forma del carácter debe ser examinado utilizando el equivalente de POINT x,y en código máquina para ver si su nueva posición está a cero antes de volver a dibujar. Así, si un carácter tiene un tamaño de  $16 \times 16$  pixels, entonces aunque sólo se examinen los pixels de las posiciones externas habría que hacer cerca de 100 llamadas a la rutina POINT antes de mover el carácter. Todo ello requiere tiempo, y por lo tanto la velocidad del movimiento se reduce drásticamente. Sin embargo, en el movimiento de fondo es posible mover un pixel cada vez mientras los atributos permanezcan fijos (si no es así no se puede mover un pixel).

El programa de demostración "PIXEL DEMO" se dispone en cinta y es nuestro primer programa de juegos de reflejos. Demuestra la lentitud del movimiento de pixel y también el hecho de que hay muchos parpadeos debido al movimiento del fondo. El balón debe ser sacado de la pantalla durante el movimiento de fondo y se debe hacer una prueba mediante POINT antes de volver a dibujarlo. Este tiempo de retardo es perceptible pero se ha conseguido hacerlo lo más corto posible mediante la elección del balón, para el cual sólo se necesita la línea externa; de esta forma reducimos al mínimo el número de veces que hay que examinar POINT y el número de llamadas a PLOT. Los

principios de actuación son los mismos que en el movimiento de cuadros de carácter, pues las posiciones x,y de PLOT se almacenan como datos, se actualiza y examina un conjunto duplicado de datos actualizados, se borra el balón utilizando OVER 1, y se vuelve a dibujar utilizando los datos duplicados. Observe que el balón puede moverse fuera de la parte izquierda y derecha de la pantalla en forma de enrosque. De hecho esto es necesario para eludir las flechas de la línea de abajo.

Para aquellos con aparatos en color hay que indicar que el programa está en blanco y negro porque si se empleasen más de dos colores, el balón podría cambiar de color al moverse de un atributo de cuadro a otro; quizás esto no importe, porque también sale bien en un aparato en blanco y negro.

El programa muestra los principios anteriores de presentación de pantalla rodante hacia la izquierda y hacia la derecha de forma que cada carácter es rotado a la izquierda con RLA, o hacia la derecha con RRA y, si se transfiere un bit al CARRY (acarreó) este es recogido por el byte del carácter adyacente. Este se tratará de nuevo en el Capítulo 9.

Para mostrar el rodamiento descendente con pixel, el siguiente programa (programa 6.20) dibuja una onda sinusoidal y una línea en la parte superior de la pantalla; después la rodará hacia abajo hasta que terminamos con una "carretera" negra suavemente ondulada. Esto puede utilizarse en un programa de carreras de coches. También muestra la forma de poner "steps" no enteros en un lazo (ver Capítulo 7).

10 FOR a=0 TO 4*PI STEP PI/44	SUB a-4*PI	
PLOT 100+SIN a*25,175: DRAW 100,0		
30 REM LLAMADA DESPLAZAMIENTO	23777 EF	rst 40
HACIA ABAJO DE UN PIXEL	defb 3 56	
40 NEXT a	EXAMINAR 1er BYTE DE MANTISA	
org 23760	PARA -ve RESPUESTA	
SET a=0 ( en la calculadora )	23780 23	inc hl
	23781 7E	ld a,(hl)
23760 EF	23782 CB 7E	bit 7,a
defb 160 56	23784 20 04	jr nz,L2
GUARDAR valor a en DATA	BORRAR CALCULADORA si -ve	
L1	23786 CD F1 2B	call 11249
23763 11 35 5D		
23766 01 05 00	VOLVER AL BASIC	
23769 ED B0		
	23789 C9	ret
PILA 4*PI ( en calc )	BORRAR CALCULADORA	
23771 EF		
defb 163 56		
23774 34	L2	
23775 34	23790 CD F1 2B	call 11249
23776 34	23793 CD 3A 5D	call ROLL

# APILAR DATO a

23796 CD 24 5D call PILA

100+SIN a\*25=x

23799 EF rst 40

sine  
deff b 31  
stack 25  
deff 52 53 72  
mult  
deff b 4  
stack 100

deff 52 55 72  
add  
deff b 15  
stack 175 (=y)  
deff 52 56 47 56

LLAMADA PLOT x,y

23813 CD DC 22 call 8924

GUARDAR H'L'

23816 D9 exx  
23817 E5 push hl  
23818 D9 exx

LD B,0 LD C,100

23819 01 64 00 ld bc,100

LD DE, +ve +ve

23822 11 01 01 ld de,257

LLAMADA DIBUJAR

23825 CD BA 24 call 9402

RECUPERAR H'L'

23828 D9 exx  
23829 E1 pop hl  
23830 D9 exx

APILAR DATO a

23831 CD 24 5D call PILA

a+PI/44  
23834 EF rst 40  
stack PI/2  
deff b 163

stack 22  
deff 52 53 48  
divide  
deff b 5  
add  
deff b 15  
end calc routine  
deff b 56  
23842 18 AF jr L1

# APILAR un SUB

PILA

23844 21 35 5D ld hl,DATA  
23847 ED 5B 65 5C ld de,(23653)  
23851 01 05 00 ld bc,5  
23854 ED B0 ldir  
23856 ED 53 65 5C ld (23653),de  
23860 C9 ret

GUARDAR DATO

DATA  
deff 0 0 0 0

DESPLAZAR HACIA ABAJO

DESPLAZAR  
23866 06 20 ld b,32  
23868 21 A0 57 ld hl,22432  
L3  
23871 7E ld a,(hl)  
23872 F5 push af  
23873 23 inc hl  
23874 10 FB djnz L3  
23876 21 BF 56 ld hl,22207  
23879 11 BF 57 ld de,22463  
23882 06 06 ld b,6  
23884 CD 86 5D call ABAJO  
23887 21 E0 4F ld hl,20448  
23890 11 00 50 ld de,20480  
23893 01 20 00 ld bc,32

23896 ED B0 ldir  
23898 21 FF 4E ld hl,20223  
23901 11 FF 4F ld de,20479  
23904 06 08 ld b,8  
23906 CD 86 5D call ABAJO  
23909 21 E0 47 ld hl,18400  
23912 11 00 48 ld de,18432  
23915 01 20 00 ld bc,32  
23918 ED B0 ldir  
23920 21 FF 46 ld hl,18175  
23923 11 FF 47 ld de,18431  
23926 06 08 ld b,8  
23928 CD 86 5D call ABAJO  
23931 06 20 ld b,32  
23933 21 1F 40 ld hl,16415  
L4  
23936 F1 pop af  
23937 77 ld (hl),a  
23938 2B dec hl  
23939 10 FB djnz L4  
23941 C9 ret

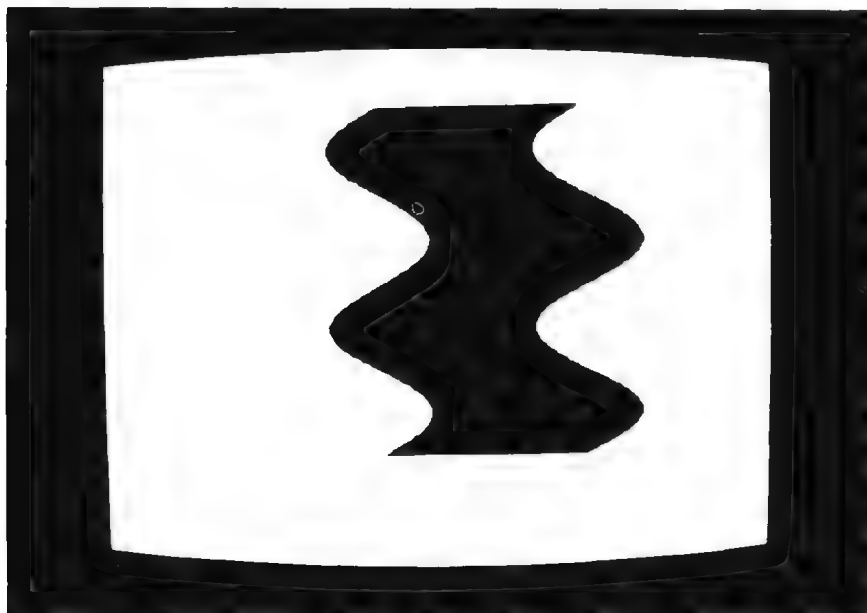
ABAJA

23942 C5 push bc  
23943 06 07 ld b,7  
L5  
23945 C5 push bc  
23946 E5 push hl  
23947 E5 push hl  
23948 01 20 00 ld bc,32  
23951 ED B8 lddr



23953 D1	pop de	23967 E5	push hl
23954 E1	pop hl		
23955 25	dec h	23968 E5	push hl
23956 C1	pop bc	23969 01 20 00	ld bc,32
23957 10 F2	djnz L5	23972 ED B8	lddr
23959 7C	ld a,h	23974 E1	pop hl
23960 C6 08	add a,8	23975 D1	pop de
23962 67	ld h,a	23976 25	dec h
23963 7D	ld a,l	23977 C1	pop bc
23964 D6 20	sub 32	23978 10 DA	djnz ABAJO
23966 6F	ld l,a	23980 C9	ret

## Programa 6.20



La rutina PIXEL DOWN ROLL que hay al final del programa 6.20 ha sido escrita de nuevo para que usted pueda ver fácilmente los procedimientos necesarios. Puede reescribirse en las líneas del programa 6.19C para hacer rodar en forma descendente con pixel una ventana, convirtiéndolo al mismo tiempo en "independiente de la posición" y usando menos memoria. En su forma actual utiliza 115 bytes (+ 32 bytes para almacenar la línea inferior), pero reescrita como el programa 6.20A utiliza sólo 44 bytes (+ 1 para almacenar el byte de la columna inferior).

org 23760		23771 25	dec h
CAPITULO 6		23772 06 23	ld b,35
PROGRAMA 20A		L1	
VENTANA RODANTE HACIA ABAJO		23774 E5	push hl
		23775 7E	ld a,(hl)
EJEMPLO		23776 12	ld (de),a
		23777 D1	pop de
COLUMNAS 4 A 10		23778 7C	ld a,h
LINEAS 4 A 39		23779 25	dec h
		23780 E6 07	and 7
REM LINEA 1/COLUMNA 1=PARTE		23782 20 0A	jr nz,L3
SUPERIOR LH.		23784 7D	ld a,l
		23785 D6 20	sub 32
23760 11 83 46	ld de,18051	23787 6F	ld l,a
LDB, Columnas		23788 38 04	jr c,L3
		23790 7C	ld a,h
23763 06 07	ld b,7	23791 C6 08	add a,8
L2		23793 67	ld h,a
23765 C5	push bc	L3	
		23794 10 EA	djnz L1
23766 D5	push de	23796 F1	pop af
23767 1A	ld a,(de)	23797 12	ld (de),a
23768 F5	push af	23798 D1	pop de
23769 D5	push de	23799 C1	pop bc
23770 E1	pop hl	23800 13	inc de
		23801 10 DA	djnz L2
		23803 C9	ret

LDB, Líneas-1

## Programa 6.20A

En el registro B se guarda el número de columnas que se van a hacer rodar (antes de L2) y también se guarda en el registro B el número de líneas - 1 (antes de L1). El registro DE se carga con la dirección de la pantalla del byte de la primera columna inferior y se calcula utilizando:

LET A = LINE (0 es la primera línea)

LET B = INT (A/64)

LET C = A - (b \* 64)

LET D = INT (C/8)

LET E = C - (D \* 8)

LET F = COLUMN (0 es la primera línea)

ADDRESS = 16384 + (1824 \* B) + (256 \* E) + (32 \* (D - 1)) + F

EXAMPLE LINE 38: COLUMN 3

A = 38: B = 0: C = 38: D = 3: E = 6: F = 3

ADDRESS = 16384 + (1824 \* 0) + (256 \* 6) + (32 \* 4) + 3 = 18051

Como esta fórmula es más bien larga, puede escribir un programa en código máquina para que haga el cálculo. Empiece guardando el número LINE/COLUMN en el par de registros HL y termine con la dirección de pantalla en DE para que así

pueda ir directamente a la rutina ROLL (alternativamente puede tener una tabla que muestre las direcciones individuales de las líneas/columnas para el mapa de memoria de la pantalla).

¿Por qué no intenta escribir tal rutina? Podría serle útil para muchos programas, por ejemplo, para sacar caracteres directamente en la pantalla con una instrucción POKE. Si se atasca, el programa 6.20B le ofrece un método, y utiliza sólo 29 bytes (serían 28 bytes si LINE/COLUMN se hubiesen almacenado en HL utilizando LD HL, 034Fh). La he unido al programa 6.20A.

Estudie este programa hasta que crea que sabe como funciona, y conseguirá tener mejor conocimiento del fichero de visualización.

org 23760		LD B, Columnas	
CAPITULO 6			
PROGRAMA 20B		23789 06 07	ld b,7
		L2	
VENTANA RODANTE HACIA ABAJO		23791 C5	push bc
		23792 D5	push de
EJEMPLO		23793 1A	ld a,(de)
		23794 F5	push af
COLUMNAS 3 A 9		23795 D5	push de
LINEAS 56 A 79		23796 E1	pop hl
		23797 25	dec g
REM LINEA 0/COLUMNA 0=PARTE			
SUPERIOR LH		LD B, Lineas-1 !	
LD H, Línea		23798 06 17	ld b,23
		L1	
23760 26 4F	ld h,79	23800 E5	push hl
		23801 7E	ld a,(hl)
LD L,Columna		23802 12	ld (de),a
		23803 D1	pop de
23762 2E 03	ld l,3	23804 7C	ld a,h
23764 11 00 40	ld de,16384	23805 25	dec h
23767 7C	ld a,h	23806 E6 07	and 7
23768 E6 07	and 7	23808 20 0A	jr nz,L3
23770 B2	or d	23810 7D	ld a,l
23771 57	ld d,a	23811 D6 20	sub 32
23772 7C	ld a,h	23813 6F	ld l,a
23773 E6 C0	and 192	23814 38 04	jr c,L3
23775 1F	rra	23816 7C	ld a,h
23776 1F	rra		
23777 1F	rra	23817 C6 08	add a,8
23778 82	add a,d	23819 67	ld h,a
23779 57	ld d,a	L3	
23780 7C	ld a,h	23820 10 EA	djnz L1
23781 E6 38	and 56	23822 F1	pop af
23783 17	ria	23823 12	ld (de),a
23784 17	ria	23824 D1	pop de
23785 5F	ld e,a	23825 C1	pop bc
23786 7D	ld a,l	23826 13	inc de
23787 83	add a,e	23827 10 DA	djnz L12
23788 5F	ld e,a	23829 C9	ret

## Programa 6.20B

Estos dos programas para el movimiento de pixel (programa 6.20 y 6.20B) muestran los métodos de rodamiento del fondo a izquierda/derecha y hacia abajo. Le dejo a usted que haga programas para UP ROLL (rodar hacia arriba) y SCROLL. Debe recordarse que en código máquina no hay ningún método perfecto. Cada persona tiene su propia forma particular de hacer las cosas, de escoger los registros, y de decidir si utiliza instrucciones “push/pop” o “store” para guardar un valor, etc., y tan pronto como el programa funcione de acuerdo con lo que uno desea entonces está bien. Muchos de los programas hasta ahora tratados podrían volverse a escribir para ahorrar memoria o reducir el tiempo de ejecución, pero entonces no serían tan fácil de seguir por un programador principiante.

Hay una rutina en ROM más que es útil y que no se dispone en el BASIC, que es CROLL n líneas con atributos, donde n debe ser mayor o igual a dos (pero no mayor que 24, por supuesto).

El programa 6.21 muestra la rutina que se está usando para borrar las líneas desde la 16 a la 24 inclusive (nueve líneas) llamando a la rutina nueve veces. Observe que el número de líneas que se van a visualizar se cuenta desde la línea 24. Puede usarse esta rutina en vez de la nuestra, escrita en el programa “MOVEDEMO” para el usuario. El registro contiene el número de líneas - 1 que se van a sacar, y después se hace una llamada a la 3584. La línea inferior se sobreimprime con espacios utilizando colores de ATTRP.

org 23760	
23760 06 009	ld b,9
L1	
23762 C5	push bc
23763 06 08	ld b,8
23765 CD 00 0E	call 3584
23768 C1	pop bc
23769 10 F7	djnz L1
23771 C9	ret

## Programa 6.21

Finalmente mostramos una rutina para desplazar una pantalla a la izquierda (o derecha) medio carácter cada vez. Este programa, aunque corto, es bastante difícil de entender, y utiliza la instrucción de rotación decimal a la izquierda (o rotación decimal a la derecha). Esta instrucción lleva consigo el manejo de nibbles del registro A y que los contenidos de la dirección se coloquen en el par de registros HL. RLD transfiere los bits 0 a 3

(HL) a los bits 7 a 4 (HL); los bits 7 a 4 (HL) a los bits 0 a 3, A; y los bits 0 a 3, A a los bits 0 a 3 (HL). No afecta a ninguno de los "flags" (bits de estado). En el programa 6.22 puede ver que se utiliza esta instrucción en cada línea de pantalla empleando el registro A para almacenar el nibble de la derecha y para colocarlo en el siguiente carácter del nibble de la izquierda. El primer carácter del nibble de la izquierda se queda en blanco porque el registro A contiene un cero al comienzo de cada línea. Al final de cada línea, el registro A contiene el nibble de la izquierda que será desplazado fuera de la pantalla. Esto se utiliza para sumarlo al byte de comienzo de línea para producir un efecto de enrosque. Si quiere un cambio de fondo continuo puede guardar el nibble de reemplazamiento de la derecha en forma de "data" para sumarlo al byte de la derecha, en vez de utilizar el nibble desplazado de la pantalla, como antes se ha indicado. Dejaré que usted escriba la rutina RIGHT ROLL (al comienzo del fichero de visualización hay una indicación-).

org 23760  
CAPITULO 6  
PROGRAMA 22

RUTINA DE DESPLAZAMIENTO A IZQ.  
DE MEDIO CUADRO DE CARACTER

23760 F3                      di

Empezar al final de la pantalla

23761 21 FF 57              ld hl,22527

N.º de líneas de pantalla

23764 06 C4                ld b,196

L1

23766 C5                    push bc

Reinicializar a=0

23767 AF                    xor a

N.º de bytes/línea

23768 06 20                ld b,32

Guardar comienzo de línea

23770 E5                    push hl

Instrucción rodar a izquierda  
un nibble a (hl)

L2

23771 ED 6F                rld

23773 2B                    dec hl

23774 10 FB                djnz L2

Obtener comienzo de línea en de

23776 D1                    pop de

Guardar actual pos. de pant.

23777 EB                    ex de, hl

Sumar nibble LH  
a byte RH

23778 86                    add a,(hl)

23779 77                    ld (hl),a

Recuperar pos. de pant. en hl

23780 EB                    ex de,hl

23781 C1                    pop bc

23782 10 EE                djnz L1

23784 FB                    ei

23785 C9                    ret

## Programa 6.22

# 7 MUSICA Y EFECTOS DE SONIDO

En este capítulo estudiaremos las formas de mejorar sus gráficos animados con sonidos interesantes y “música”. Pero primero debemos echar una ojeada a la forma que tiene la Spectrum de almacenar números en forma de coma flotante, y también veremos cómo convertir un número en forma de coma flotante en su forma comprimida. Si sus matemáticas están abandonadas no se preocupe demasiado por la teoría, porque la Spectrum puede ayudarle. Se trata sencillamente de que cualquier número, decimal, entero, positivo o negativo puede expresarse en forma de cinco-bytes utilizando un procedimiento de conjunto para definir cada byte. Al primer byte se le llama *exponente*, y a los cuatro restantes *mantisa*.

La conversión de un número “normal” a su forma de cinco-bytes requiere los cuatro pasos siguientes:

1. Expresar el número en su forma *binaria*.  
Por ejemplo 11.375 sería 1011.011  
Recuerde que  $0.1$  en binario =  $\frac{1}{2}$   
 $0.01$  en binario =  $\frac{1}{4}$   
etc.
2. Calcular el *exponente*.  
Mover la coma decimal a la izquierda o a la derecha hasta que esté a la izquierda del primer 1 en el número binario, y contar el número de movimientos realizados.  
El EXPONENTE será  $128 +$  el número de movimientos realizados.  
Si los movimientos son hacia la izquierda, entonces los “movimientos” son positivos. Si los movimientos son hacia la derecha, entonces los “movimientos” son negativos. En nuestro ejemplo, el exponente =  $128 + 4 = 132$  y nos quedamos con .1011011.
3. Cambiar el bit a la derecha de la coma decimal para indicar el signo del número decimal.  
Si el número era positivo entonces el bit se pone a cero.  
Si el número era negativo, entonces el bit se pone a uno.  
En nuestro ejemplo era positivo y se transforma en .0011011.

#### 4. Encontrar la *mantisa*.

Descartar la coma decimal y dividir el número binario restante en cuatro bytes, añadiendo ceros si es necesario, para componer 32 bits.

00110110 00000000 00000000 00000000

y convertirse entonces cada byte otra vez a su número decimal.

Y así, se puede ver nuestro ejemplo que

11.375 = 132 45 0 0 0

Si ahora está completamente confundido, como le he dicho, la Spectrum puede ayudarle. La Spectrum tiene un método útil (útil para nosotros) para almacenar variables simples, inmediatamente después del código de las variables, en su forma de coma flotante de cinco-bytes. De modo que todo lo que necesitamos hacer es escribir un programa corto para ingresar (con ENTER) un número, y después recogerlo (con PEEK) en una variable para encontrar su forma de cinco bytes.

```
5 GO TO 5000
10 REM PROGRAMA PARA IMPRIMIR
CUALQUIER NUMERO EN FORMA DE
COMA FLOTANTE
```

```
20 PRINT AT 3,0; "EL SIGUIENTE PRO-
GRAMA CORTO IMPRIMIRA CUALQUIER
NUMERO, DECIMAL, ENTERO O NEGA-
TIVO EN FORMA DE COMA FLOTANTE.
```

```
   A LOS NUMEROS ENTEROS SE
LES SUMA EL DECIMAL 0.000000000001
PARA CONVERTIRLOS A UN NUMERO DE
COMA FLOTANTE PERO NO AFECTA A
SU VALOR.
```

```
25 PRINT " " "LOS NUMEROS EN FORMA
COMPRIMIDA TAMBIEN PUEDEN USARSE
CON EL LITERAL DE CALCULADORA 34H
(52d)
```

```
40 RETURN
100 INPUT "NUMERO:" TAB 9;n
200 LET n$="NUMERO "
300 LET b$="COMA FLOTANTE : "
400 LET c$="FORMA COMPRIMIDA:"
```

```
500 LET d$="□"
1003 IF N <> 0 THEN GO TO 1010
1005 PRINT N$=; TAB 9; d$; PAPER 6;0
'b$; TAB 9; d$; "0 ";d$;"0 ";d$;"0
";d$;"0 ";d$;"0"
1006 PRINT c$;TAB 9;d"0 "d$;"
176";d$;"0"
1007 GO TO 100
1010 LET N=N+.000000000001
1020 PRINT n$;TAB 9;d$; PAPER 6;N
1025 PRINT b$;
```

```

1030 LET X=PEEK 23627+256*PEEK 23
628
1040 FOR A=1 TO 5
1050 PRINT TAB 5+4*A;d$;PEEK (X+A);
1060 NEXT A
1065 PRINT '
1070 DIM a(5)
1075 LET q=0
1076 PRINT c$;
1080 FOR b=5 TO 1 STEP -1
1085 LET a(b)=PEEK ((PEEK 23627+2)
56*PEEK 23628)+b)
1086 IF b=2 THEN GO TO 1100
1090 IF q=0 AND a(b)=0 THEN NEXT b
1100 LET q=q+1
1110 NEXT b
1120 LET q=q-2
1125 LET x=(a(1)-80)
1126 IF x>=64 THEN GO TO 2000
1130 LET e=x+(q*64)
1140 PRINT TAB 9;d$;e;
1150 FOR b=1 TO q+1
1160 PRINT TAB 9+b*4; d$;a(1+b);
1170 NEXT b
1180 PRINT
1190 GO TO 100
2000 PRINT TAB 9; d$;q*64;TAB 13;d
$a(1)-80;
2010 FOR b=1 TO q+1
2020 PRINT TAB 13+b*4;d$a(1+b);
2030 NEXT b
2040 PRINT : GO TO 100
5000 DATA 24,79,5,2, "PARE LA CINTA"
5100 DATA 15,167,7,4, "NUMEROS"
5200 DATA 15,130,2,4, "_____"
5250 DATA 24,15,1,2, "PULSE CUALQUIER
TECLA"
5500 DATA 31,170,1,2,"INSTRUCCIONES"
6000 BORDER 45: PAPER 6: INK 0: CL S
6010 LET n=0
6050 RESTORE 5000
6100 FOR a=1 TO 4
6200 INK a: GO SUB 8000
6300 PAUSE 50: NEXT a
6400 PAUSE 0: CLS
6420 INK 2: GO SUB 8000
6450 INK 0: PAPER 7: GO SUB 10
7000 RESTORE 5250: INK 2: GO SUB
8000
7005 IF INKEY$<>" " THEN GO TO 7005
7006 IF INKEY$=" " THEN GO TO 7006
7010 CLS : GO TO 100
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 32334: LOAD "udgs" CODE
USR "a",168: LOAD ""CODE : GO TO
5000
9999 SAVEW "NUMBER" LINE 998: SAVE
"udgs" CODE USR "a",168: SAVE "LARGE"
CODE 32335,265
<336 S.A.Nicholls

```

## Programa "Números"



Este programa está disponible en cinta y tiene colores, como parte de las series n\$ b\$ c\$ y d\$.

Después de RUN, el programa imprimirá los cinco bytes de coma flotante, y la forma comprimida de cualquier número. La forma comprimida se discutirá más tarde. Observe que el 0 es tratado como un caso especial en el cual todos los bytes son 0, y también que 0,5 y  $\frac{1}{2}$  dan resultados distintos.

Hasta ahora se preguntará qué tiene que ver esto con la música y sonido. La respuesta es que la orden BEEP del BASIC hace uso de la pila de la calculadora, y cualquier número almacenado en la pila debe ponerse en formato de cinco-bytes.

Ahora podemos buscar la forma de convertir en código máquina la instrucción BEEP duración, tono. Este método consiste en apilar el valor de la duración y del tono, en formato de cinco-bytes, en la parte superior de la pila de la calculadora y llamar después a la rutina BEEP desde la dirección 03F8h (1016 d).

Si tomamos, por ejemplo, BEEP.1,11, entonces, utilizando el programa NUMBERS podemos convertir .1 y 11 en su forma de cinco-bytes:

```
.1 = 125 76 204 204 204
11 = 132 48 0 0 0
```

Ahora debemos encontrar la manera de apilar esos diez bytes consecutivamente en la pila de la calculadora. Un método podría ser encontrar la dirección de comienzo desde la cual los DATA deben ser transferidos mediante PEEK a la variable STACKEND del sistema, y utilizar después una instrucción LDIR tal y como se muestra en el programa 7.1

org 23760		23772 ED 53 65 5C	ld (23653),de
CAPITULO 7			
PROGRAMA 1		LLAMADA BEEP	
BEEP .1,11			
ENCONTRAR STACKEND		23776 CD F8 03	call 1016
		23779 C9	ret
23760 ED 5B 65 5C	ld de,(23653)	DATA	
23764 21 E4 5C	ld hl,DATA	0.1 (forma de 5 bytes)	
23767 01 0A 00	ld bc,10	defb 125 76 204 204 204	
23770 ED B0	ldir		
		11 (forma de 5 bytes)	
ACTUALIZAR STACKEND		defb 132 48 0 0 0	

## Programa 7.1

El RESET es necesario para volver a cargar STACKEND con la nueva dirección de la parte superior de la pila. Recuerde

que LDIR termina con HL y DE conteniendo la *dirección* del último byte transferido + 1. Cuando se llama a la rutina BEEP desde la dirección 03F8 ésta toma los dos valores superiores de la pila y opera con ellos para obtener la duración y el tono requeridos. Se reinicializa la pila para volver al BASIC.

Otro método para apilar números enteros es utilizar LDA,n y la subrutina CALL STACK A de la dirección 2D28h (11560 d) para números entre 0 y 255, ó LDBC,nm y la subrutina CALL STACK BC de la dirección 2D2Bh (115 63 d) para números entre 0 y 65535. Entonces el programa 7.1 puede escribirse como el programa 7.2.

org 23760	23772 ED 53 65 5C	ld (23653),de
CAPITULO 7	23776 3E 0B	ld a,11
PROGRAMA 2		
BEEP .1,11	APILAR valor a	
	23778 CD 28 2D	call 11560
ENCONTRAR STACKEND	CALL BEEP	
23760 ED 5B 65 5C	ld de,(23653)	
23764 21 E9 5C	ld hl,DATA	23781 CD F8 03
23767 01 05 00	ld bc,5	23784 C9
23770 ED B0	ldir	call 1016
		ret
	DATA	
	0.1 (forma de 5 bytes)	
ACTUALIZAR STACKEND	deff 125 76 204 204 204	

## Programa 7.2

Por último, para aquellos que deseen algo más que los números de coma flotante, podemos convertirlos en su *forma comprimida* y utilizar el literal 34h(52 d) de la calculadora para apilar datos en esta forma comprimida.

Las etapas para convertir los números de cinco-bytes son las siguientes:

1. Empezar con el quinto byte, sacar todos los números cero hasta que encontremos un número distinto de cero o hasta que lleguemos al segundo byte.
2. El cociente es entonces el número de los bytes restantes *menos 1*.
3. Restar 50h (80d) del exponente y, si el resultado es mayor o igual a 40h (64d) entonces ir a la instrucción 6.
4. Cambiar el exponente a (exponente - 80d) + (cociente \* 64d).
5. Ir a la instrucción 7.
6. El exponente tiene ahora dos bytes. El primer byte es el cociente \* 64d; el segundo byte es el exponente - 80d.
7. Colocar los bytes que quedan, detrás de los bytes del exponente modificados.

Para volver a convertir un número comprimido en su forma de cinco-bytes son necesarias las siguientes etapas:

1. Dividir el primer byte por 64d.
2. Si queda un resto, entonces el exponente será 80d + el resto.
3. Si no hay resto entonces el exponente será 80d + el segundo byte.
4. El cociente es utilizado para encontrar cuantos bytes extra siguen al exponente.

El número de bytes extra es el cociente + 1.

Por ejemplo  $11 = 5248$  en su forma comprimida:

$52 \text{ dividido por } 64 = 0 \text{ con } 52$

de resto; cociente = 0

bytes posteriores = 1.

Como hubo resto, entonces el exponente =  $52 + 80 = 132$

1 byte posterior especificado = 48

Rellenar los 5 bytes con 0 0 0

Por lo tanto,  $52 \ 48 = 132 \ 48 \ 0 \ 0 \ 0$

De nuevo, si todo esto es excesivo para usted por ahora, entonces el programa NUMBERS hará todas las conversiones por usted. Utilizando números en forma comprimida, BEEP .1,11 puede convertirse al código máquina, como se muestra en el programa 7.3.

org 23760  
CAPITULO 7  
PROGRAMA 3

BEEP .1,11

UTILIZAR CALCULADORA

23760 EF                      rst 40

APILAR 0,1 (FORMA COMP.)

defb 52 237 76 204 204 204

STACK 11 (FORMA COMP.)

defb 52 52 48

FIN CALC

defb 56

LLAMADA BEEP

23771 CD F8 03

23774 C9

call 1016

ret

### Programa 7.3

En el programa 7.3 RST 28h (40d) llama a la rutina CALCULATOR, y defb 38h (56d) es el literal END CALC.

El FANFARE BEEP para el programa "CROSS" es una combinación de los tres métodos, a saber, almacenamiento de los números como DATA, utilización de STACK A y utilización del literal 34h (52d) para apilar números en forma comprimida. En este programa se tiene la suerte de que los bytes de la mantisa de los decimales .1, .8 y 0.5 en forma comprimida son iguales,

por lo que sólo se necesita almacenar en DATA el byte del exponente.

1 REM CAPITULO 7		VALOR	
2 REM PROGRAMA 4		defb 0 76 204 204 204 56	
REM FANFARE BEEP		23780 E1	pop hl
10 FOR a=1 TO 8		23781 7E	ld a,(hl)
20 READ b,c		23782 E5	push hl
30 BEEP b,c		23783 CD 28 2D	call 11560
50 NEXT a		23786 CD F8 03	call 1016
60 DATA .1,11.,1,11.,8,16.,05,		23789 E1	pop hl
11.,05,16.,05,11.,05,16,1,20		23790 C1	pop bc
org 23760		23791 23	inc hl
23760 06 07	ld b,7	23792 10 E3	djnz LAZO
23762 21 00 5D	ld hl,DATA	23794 3E 01	ld a,1
LAZO		23796 CD 28 2D	call 11560
23765 7E	ld a,(hl)	23799 3E 14	ld a,20
23766 32 DE 5C	ld (VALOR),a	23801 CD 28 2D	call 11560
23769 23	inc hl	23804 CD F8 03	call 1016
23770 C5	push bc	23807 C9	ret
23771 E5	push hl	DATA	
23772 EF	rst 00	defb 237 11 237 11 240 16 236	
defb 52		defb 11 236 16 236 11 236 16	

## Programa 7.4

Ya he mostrado como convertir cualquier orden BEEP del BASIC al código máquina. Esto está bien para hacer música, pero de esta forma no obtendremos efectos de sonido decentes. Para este tipo de sonidos necesitaremos entrar en la rutina BEEP en un punto diferente, 03B5h (949d). Después, los valores de la duración y del tono que han sido sacados de la pila son manipulados, y se termina con el “tono” en el registro HL y la “duración” en el registro DE. De esta forma se podría escribir un programa para conseguir un tono con cadencia incrementando el valor almacenado en HL y llamando a la rutina BEEP repetidamente mediante un lazo “FOR/NEXT”. Podemos realizar esto como se muestra en el programa 7.5.

org 23760		LLAMADA BEEP	
CAPITULO 7		23770 CD B5 03	call 949
PROGRAMA 5		23773 C1	pop bc
SONIDO 1		23774 E1	pop hl
		23775 23	inc hl
23760 06 FF	ld b,255	23776 10 F3	djnz LAZO
23762 21 01 00	ld hl,1		
LAZO		23778 C9	ret
23765 11 01 00	ld de,1		
23768 E5	push hl		
23769 C5	push bc		

## Programa 7.5

Cuando ejecute el Programa 7.5 observará que según varíe el tono así cambiará gradualmente la duración. Esto es así debido a que de hecho, la duración es función de los valores de los registros HL y DE. Para igualar la duración debemos decrementar el valor de DE según se incremente el valor de los registros. HL. Así llegamos al Programa 7.6.

org 23760		23770 C5	push bc
CAPITULO 7		23771 CD B5 03	call 949
PROGRAMA 6		23774 C1	pop bc
		23775 E1	pop hl
SONIDO 2		23776 D1	pop de
		23777 23	inc hl
23760 06 64	ld b,100	23778 1B	dec de
23762 21 01 00	ld hl,1	23779 10 F3	djnz LAZO
23765 11 65 00	ld de,101		
LAZO			
23768 D5	push de	23781 C9	ret
23769 E5	push hl		

## Programa 7.6

Tenemos un último método para producir sonido con la Spectrum, que consiste en enviar “señales” directamente al altavoz conmutándolo a alta o a baja de la misma manera que lo hace la rutina de la ROM. Este método parece poco útil para obtener notas musicales, ya que necesitaríamos conocer la frecuencia de conmutación de alta/baja para cada nota, y de hecho podría ser una repetición de la rutina de la ROM. Sin embargo, se utiliza en programas de juegos para producir un sonido de tipo “ruido blanco” para explosiones; etc.

La instrucción para comunicar con el altavoz es OUT (254),A. Esta instrucción no sólo controla el altavoz, sino también el color de los bordes de la siguiente forma:

1. Si el bit 0,A está a uno, entonces se activa el azul.  
Si el bit 0,A está a cero, entonces se desactiva el azul.
  2. Si el bit 1,A está a uno, entonces se activa el rojo.  
Si el bit 1,A está a cero, entonces se desactiva el rojo.
  3. Si el bit 2,A está a uno, entonces se activa el verde.  
Si el bit 2,A está a cero, entonces se desactiva el verde.
  4. Si el bit 4,A está a uno, entonces el altavoz está en “alta”.  
Si el bit 4,A está a cero, entonces el altavoz está en “baja”.
- Como puede ver, si el bit 4 cambiase de cero a uno a intervalos regulares, entonces se podría producir una nota dependiente de este intervalo.

Para obtener un ruido blanco auténtico necesitaríamos conmutar con un intervalo aleatorio, pero ello requeriría una rutina larga para obtener el número aleatorio y examinar el bit 4,A

cada vez, lo que originaría una pausa “larga” entre cada conmutación. Por lo tanto debemos encontrar un método rápido de obtención de “números aleatorios”. Esto no es tan difícil como parece; a nosotros sólo nos interesa el estado del bit 4,A, y para nuestros propósitos los bytes de la ROM son suficientemente aleatorios. Con esto en mente podemos escribir un programa que busque en cada dirección de la ROM de forma sucesiva, y examine el estado del bit 4 del número contenido en cada dirección para conmutar el altavoz a alta o a baja de acuerdo con él.

org 23760		23771 00	nop
23760 21 00 00	ld hl,0	23772 00	nop
L1		23773 D3 FE	out (254),a
23763 7E	ld a,(hl)	23775 00	nop
23764 00	nop	23776 23	inc hl
23765 D3 FE	out (254),a	23777 7C	ld a,h
2376700	nop	23778 FE 3C	cp 60
23768 00	nop	23780 20 ED	jr nz,L1
23769 D3 FE	out (254),a	23782 C9	ret

## Programa 7.7

Hay varios puntos a observar en el programa 7.7.

1. Las tres instrucciones OUT (254),A y NOPs producen un retardo. Usted puede añadirle o quitarle más y escuchar el efecto.
2. La longitud de sonido es gobernada por la instrucción CP 60. Si se redujese a CP 40, por ejemplo, la longitud del sonido se reduciría de forma similar. Cualquier valor inferior a 60 se ejecuta en un área de ROM que no contiene números aleatorios. Pruebe y vea qué sucede.
3. El color de los bordes también queda afectado temporalmente según cambian los bits 0-2 de forma aleatoria. Para que los bordes permanezcan con el mismo color necesita reemplazar los bits 0-2 por los bits 3-5 del valor que contiene la variable BORDER del sistema.

Si quiere experimentar con este método de obtención de música recuerde que se accede 50 veces por segundo a la rutina de interrupción de la ROM y ello alterará su frecuencia, por lo que necesitará deshabilitarla (DI) antes de su rutina, y habilitarla (EI) después. Esto es lo que hace la rutina de la ROM, y explica por qué no se puede producir un BREAK de un programa BASIC durante la ejecución de una orden BEEP.

Dejo el resto a su imaginación e ingenuidad para que altere los valores contenidos en los registros HL y DE al objeto de obtener efectos maravillosos y siniestros, pero para darle una idea de los sonidos que se pueden conseguir usando el método anterior, la cinta contiene el Programa SOUND que tiene cinco so-

nidos diferentes a los que se accede pulsando las teclas de la 1 a la 5. El borde parpadeante produce una dimensión extra de los sonidos 1 y 5, y es muy simple de programar.

Una última cuestión —produzca sonidos tan cortos como sea posible ya que éstos detienen todos los movimientos gráficos durante su ejecución. Los mejores sitios para los sonidos son los lazos de retardo utilizados para relentizar los movimientos gráficos.



```

10 REM ? RETURN RUN GO SUB X
PRINT THEN G TO ?] LET PRINT THEN O
TO ]]LET PRINT THEN W TO D] LET PRINT
THEN _ TO _] LET PRINT THEN g TO []
LET >? STEP £"? RETURN @ GO SUB X
THEN G>=?<>
2 OR 1d?
25>INK 2: PAPER 7:BORDER 4:CLS
30 LET x=16: LET y=100: LET w=2: LET
h=4: LET a$="PARE LA CINTA"
40 PAPER 8: RANDOMIZE USR 32393
50 PRINT #0:AT 0,3:"PULSE CUAL-
QUIER TECLA PARA EMPEZAR"
60 PAUSE 0
65 CLS
70 LET x=8: LET y=175: LET w=5: LET
h=15: LET a$="SONIDO!"
80 INK 2: RANDOMIZE USR 32393
90 PRINT AT 19,2:"Pulse teclas 1 a 5
para sonidos"

```

```

100 PRINT AT 21,3;"Con SPACE volverá
al BASIC"
110 RANDOMIZE USR 23760
120 CLS
130 LET x=24: LET y=100: LET h=5: LET
w=1: LET a$="To re RUN enter GOTO 65
140 RANDOMIZE USR 32393
150 STOP
9998 CLEAR 32334: LOAD ""CODE: GO
TO 25
9999 SAVE "SOUNDS" LINE 9998: SAVE
"LARGE"CODE 32335,265: STOP <336 @
S.A.Nicholls

```

```

Org 23760
COMIENZO
23760 01 FE F7      ld bc,63486
23763 ED 78        in a,(c)
23765 F5           push af
23766 CB 47        bit 0,a
23768 CC 07 5D     call z,BEEP1
23771 F1          pop af
23772 F5           push af
23773 CB 4F        bit 1,a
23775 CC 29 5D     call z,BEEP2
23778 F1          pop af
23779 F5           push af
23780 CB 57        bit 2,a
23782 CC 44 5D     call z,BEEP3
23785 F1          pop af
23786 F5           push af
23787 CB 5F        bit 3,a
23789 CC 5F 5D     call z,BEEP4
23792 F1          pop af

```

```

23793 F5           push af
23794 CB 67        bit 4,a
23796 CC 7B 5D     call z,REEP5
23799 F1          pop af
23800 3E 07        ld a,7
23802 CD 9B 22     call 8859
23805 01 FE 7F     ld bc,32766
23808 ED 78        in a,(c)
23810 CB 47        bit 0,a
23812 C8          ret z
23813 18 C9        jr COMIENZO
BEEp1
23815 06 32        ld b,50
L1
23817 C5           push bc
23818 21 64 00     ld hl,100
L2
23821 11 0A 00     ld de,10
23824 E5           push hl
23825 7D          ld a,1
23826 E6 07        and 7
23828 CD 9B 22     call 8859
23831 CD B5 03     call 949
23834 E1          pop hl
23835 11 0A 00     ld de,10
23838 AF          xor a
23839 ED 52        sbc hl,de
23841 7C          ld a,h
23842 B5          or 1

```



23843 20 E8	jr nz,L2
23845 C1	pop bc
23846 10 E1	djnz L1
23848 C9	ret
BEEP2	
23849 21 00 00	ld hl,0
L3	
23852 7E	ld a,(hl)
23853 00	nop
23854 D3 FE	out (254),a
23856 00	nop
23857 00	nop
23858 D3 FE	out (254),a

23860 00	nop
23861 00	nop
23862 D3 FE	out (254),a
23864 00	nop
23865 00	nop
23866 D3 FE	out (254),a
23868 00	nop
23869 23	inc hl
23870 7C	ld a,h
23871 FE 3C	cp 60
23873 20 E9	jr nz,L3
23875 C9	ret
BEEP3	
23876 06 05	ld b,5
L5	
23878 C5	push bc
23879 21 32 00	ld hl,50
L4	
23882 11 0A 00	ld de,10
23885 E5	push hl
23886 CD B5 03	call 949
23889 E1	pop hl
23890 11 32 00	ld de,50
23893 19	add hl,de
23894 7C	ld a,h
23895 FE 04	cp 4
23897 20 EF	jr nz,L4
23899 C1	pop bc
23900 10 E8	djnz L5
23902 C9	ret
BEEP4	
23903 06 0A	ld b,10
L6	
23905 C5	push bc
23906 21 00 04	ld hl,1024
L7	
23909 11 01 00	ld de,1
23912 E5	push hl
23913 CD B5 03	call 949
23916 E1	pop hl
23917 AF	xor a
23918 11 10 00	ld de,16
23921 ED 52	sbc hl,de
23923 7C	ld a,h
23924 B5	or 1
23925 20 EE	jr nz,L7
23927 C1	pop bc
23928 10 E7	djnz L6
23930 C9	ret
BEEP5	

23931 21 01 00	ld hl,1
L8	
23934 11 05 00	ld de,5
23937 E5	push hl
23938 CD B5 03	call 949
23941 E1	pop hl
23942 23	inc hl
23943 7C	ld a,h
23944 FE 02	cp 2
23946 20 F2	jr nz,L8
23948 06 FA	ld b,250
L9	
23950 C5	push bc
23951 78	ld a,b
23952 E6 06	and 6
23954 CD 9B 22	call 8859
23957 21 90 01	ld hl,400
23960 11 01 00	ld de,1
23963 CD B5 03	call 949
23966 C1	pop bc
23967 10 ED	djnz L9
23969 C9	ret

## Programa "Sound"

## 8 ATTRIBUTE, SCREEN\$ Y POINT

Como usted ya sabe, si conoce la programación de juegos en BASIC, llegará un momento en el que se necesitará encontrar un carácter que está en una posición particular de la pantalla. Esto es útil, por ejemplo, si tiene que examinar si un misil ha dado en el blanco, o si ha conseguido alunizar. Hay tres formas de hacer esto en BASIC, a saber:

Con **ATTRIBUTE** que examina INK: PAPER: FLASH: BRIGHT, es decir, el estado de un cuadro de carácter, y regresa con un número entre 0 y 255.

Con **SCREEN\$** que examina el carácter en un cuadro particular (pero sólo los caracteres comprendidos entre CODE 32 y 127); para los restantes caracteres devuelve una serie vacía).

CON **POINT** que examina el estado de un pixel de pantalla: 0 = no dibujado, 1 = dibujado.

Los tres métodos anteriores son disponibles en código máquina haciendo llamadas a rutinas ROM.

### ATTRIBUTE (línea, columna)

La rutina en código máquina —call 9603— requiere el número de línea en el registro C y el número de columna en el registro B. El valor de ATTRIBUTE después de llamar a la rutina queda almacenado en la parte superior de la pila de la calculadora. El valor de ATTRIBUTE se calcula de la siguiente forma:

Valor = 128 \* (número de FLASH) + 64 \* (número de BRIGHT) + 8 (color de PAPER) + color de INK

Puede ver en la fórmula anterior que el valor está comprendido entre 0 y 255 y así puede moverse fácilmente desde la pila al registro A utilizando la rutina "CALC VALUE TO A" —call 11733— y probarse con una instrucción CPn.

El programa 8.1 muestra el método anterior utilizado para PRINT AT 0, 20; PAPER 1; INK 7; FLASH 1; CHR\$ 144 (que ha sido redefinido como un hombre). El programa examina entonces cada posición de pantalla, mostrada por un signo > ,

hasta que encuentra el valor 143 de ATTR; después vuelve al BASIC.

org 23760		SACAR >	
CAPITULO 8		23819 11 3E 5D	ld de,DATA2
PROGRAMA 1		23822 01 04 00	ld bc,4
EXAMINAR ATRIBUTOS		23825 CD 3C 20	call 8252
ESTABLECER UDG		INC. COLUMNA	
23760 ED 5B 7B 5C	ld de,(23675)	23828 3A 40 5D	ld a,(COL)
23764 21 42 5D	ld hl,DATA3	23831 3C	inc a
23767 01 08 00	ld bc,8	23832 32 40 5D	ld (COL),a
23770 ED B0	ldir	OBTENER COLUMNA/LINEA en BC	
BORRAR PANTALLA		23835 ED 4B 3F 5D	ld bc, (LINEA)
23772 3E 02	ld a,2	LLAMADA ATTR. &	
23774 CD 01 16	call 5633	OBTENER VALOR EN A	
23777 CD 6B 0D	call 3435		
23780 3E 02	ld a,2	23839 CD 83 25	call 9603
23782 CD 01 16	call 5633	23842 CD D5 2D	call 11733
IMPRIMIR UDG		EXAMINAR SI 143	
23785 11 2E 5D	ld de,DATA 1	23845 FE 8F	cp 143
23788 01 10 00	ld bc,16	MOVERSE A SIGUIENTE POS.	
23791 CD 3C 20	call 8252	23847 20 D1	jr nz,L1
ESTABLECER OVER 1		REINICIAR OVER 0	
23794 FD 36 57 03	ld (iy+87),3		
REINICIAR COMIENZO EN 0/0		23849 FD 36 57 00	ld (iy+87),0
23798 AF	xor a	23853 C9	ret
23799 32 40 5D	ld (COL),a	IMPRIMIR DATOS UDG	
IMPRIMIR >		DATA1	
L1		defb 22 0 20 17 1 16 7 18 1	
23802 11 3E 5D	ld de, DATA2	defb 144 18 0 17 7 16 0	
23805 01 04 00	ld bc,4	PRINT > DATA	
23808 CD 3C 20	call 8252	DATA2	
RETARDO		defb 22	
23811 21 FF FF	ld hl,65535	LINEA	
L2		defb 0	
23814 2B	dec hl	COL	
23815 7C	ld a,h	defb 0 62	
23816 B5	or 1	DATOS UDG	
23817 20 FB	jr nz,L2	DATA3	
		defb 24 153 126 153 24 36 36 102	

## Programa 8.1

Cuando se ejecuta, el programa 8.1 da la apariencia de un misil moviéndose lentamente hacia un hombre, y vuelve al BASIC cuando choca con el hombre. Por supuesto, podría utilizar una rutina de choque en vez de la instrucción RETURN (con efectos de sonido). Esta forma de examinar las posiciones de pantalla para blancos, etc., está bien, pero es algo limitada porque todos los blancos deben tener el mismo valor de ATTRIBUTE.

## SCREEN\$ (línea, columna)

Al igual que con la rutina ATTR el número de línea se guarda en el registro C, y el número de columna en el registro B antes de hacer una llamada a 9528. La rutina guarda los *parámetros de la serie* en la parte superior de la pila de la calculadora en forma de cinco-bytes. El uso de la pila de la calculadora para guardar los parámetros de la serie es una buena idea puesto que hasta ahora sólo la hemos usado para guardar números en forma de cinco-bytes. Hay que decir que hay literales de calculadora para manipular series, pero en vez de poner la serie en la pila, sus "parámetros" se guardan en forma de cinco-bytes. Para poner los parámetros de la serie en la pila, el par de registros BC guarda la longitud de la serie, y el par de registros DE guarda la dirección de comienzo de la serie. Para series sencillas, el registro A almacena el valor cero. Una llamada a la rutina —call 10929— colocará los parámetros en la pila en el orden correcto. De forma similar, la rutina —call 11249— tomará el último "valor" de la pila y lo colocará en los registros A, B, C, D y E; de nuevo BC contendrá la longitud de la serie, y DE la dirección de comienzo.

El programa 8.2 muestra las rutinas que se usan para imprimir el carácter @ en la posición SCREEN 0,20; y cuando se ejecuta termina con dos @ en 0,20; y 0,21. En vez de PRINT STRING —call 8252— podríamos haber utilizado LD A,(DE) para examinar si el carácter anual está en la posición 0, 20; y después realizar las acciones oportunas.

Sin embargo, hay una limitación en la rutina SCREEN\$ del BASIC, y es que sólo encontrará caracteres comprendidos entre CODE 32 y CODE 127; para cualquier otro valor se obtendrá una serie vacía. Esto excluye la utilización de este método para encontrar UDGs, y en BASIC esto es una gran contrariedad para los programas de juegos. En código máquina esto no es problema, puesto que hay formas "rápidas" de engañar a la ROM y de colocar los UDGs en el conjunto de caracteres. La

variable del sistema CHARS-23606/7 normalmente guarda la dirección de comienzo del conjunto de caracteres -256, es decir 15360, y es utilizada por la rutina SCREEN\$ para encontrar el carácter. Podemos establecer esta variable para que apunte al comienzo de los UDGs -256 antes de que se haga la llamada a SCREEN\$ (convirtiendo UDG"A" en CODE 32, UDG"B" en CODE 33, etc.) y después reinicializarla a su valor normal. De esta forma podemos examinar los UDGs para encontrar el carácter que queremos.

org 23760		23779 3E 14	ld a,20
CAPITULO 8		23781 D7	rst 16
PROGRAMA 2		23782 3E40	ld a,64
		23784 D7	rst 16
SCREEN\$ (0,20)			
BORRAR PANTALLA		LLAMADA SCREEN\$	
23760 3E 02	ld a,2	23785 06 14	ld b,20
		23787 0E 00	ld c,0
23762 CD 01 16	call 5633		
23765 CD 6B 0D	call 3435	23789 CD 38 25	call 9528
23768 3E 02	ld a,2		
23779 CD 01 16	call 5633	LLAMADA 'VALOR A A, B, C, D, E	
IMPRIMIR EN 0,20 @		23792 CD F1 2B	call 11249
23773 3E 16	ld a,22	LLAMADA IMPRIMIR SERIE	
23775 D7	rst 16		
23776 3E 00	ld a,0	23795 CD 3C 20	call 8252
23778 D7	rst 16	23798 C9	ret

## Programa 8.2

El programa 8.3 es lo mismo que el programa 8.1 pero esta vez estamos buscando CHR\$ 144 antes de volver al BASIC.

org 23836		23846 CD F1 2B	call 11249
CAPITULO 8			
PROGRAMA 3		EXAMINAR SI ES CHR\$ (32)	
SCREEN\$ PARA UDG's		por ejemplo CHR\$ 144 (UDG 'A')	
23760 TO 23835		23849 1A	ld a,(de)
IGUAL QUE PROG. 1		23850 FE 20	cp 32
ESTABLECER CARACTS. A UDG's -256		REINICIALIZAR CARACTS. A NORMAL	
23836 2A 7B 5C	ld hl,(23675)	23852 21 00 3C	ld hl,15360
23839 25	dec h	23855 22 36 5C	ld (23606),hl
23840 22 36 5C	ld (23606),hl		
EXAMINAR SCREEN\$		RESTO DEL PROGRAMA	
		IGUAL QUE PROGRAMA 1	
23843 CD 38 25	call 9528	23847 PARA TERMINAR DATA3	

## Programa 8.3

Por supuesto, podría copiar el conjunto de caracteres en RAM y mantener a CHARS apuntando permanentemente al comienzo -256. De esta forma puede olvidarse de los UDGs y redefinir cualquier carácter que desee en la nueva serie (deje sin cambiar las letras y los números para que el texto no se altere). Este método conseguirá alrededor de 50 caracteres redefinibles si se limita a utilizar el texto con *letras mayúsculas* y un mínimo de signos de puntuación. La rutina SCREEN\$ reconocerá estos caracteres cuando aparezcan en la nueva serie de caracteres.

Hay un punto muy importante a considerar cuando utilice la calculadora para manejar series, y es que se hace uso del espacio de trabajo para guardar la serie y/o el resultado del cálculo. Si este espacio de trabajo no se borra regularmente entonces puede encontrarse con que en su programa aparece un mensaje de fuera de memoria sin razón aparente, o peor todavía, el espacio de trabajo se ejecutará en su rutina de código máquina y su programa se descontrolará. En la práctica es aconsejable borrar el espacio de trabajo cada vez que se llama a la rutina SCREEN\$. La rutina de la ROM —call 5823— borra para usted el espacio de trabajo (y también borra la pila de la calculadora). Así, el programa 8.3 debería llamar a 5823 después de LD A,(DE) pero recuerde que debe guardar el valor de A antes de hacer la llamada.

## POINT

La instrucción POINT x,y, del BASIC vuelve con un valor de 1 si el pixel x, y está establecido (es decir, dibujado) o 0 si no lo está (PLOT OVER). La rutina en código máquina —call 8910— requiere que la coordenada x se guarde en el registro C y la coordenada y en el registro B antes de hacer la llamada. Al igual que con las rutinas anteriores el resultado termina con el valor en la pila de la calculadora, y, como este valor es 0 ó 1 podemos sacarlo de la pila y ponerlo en el registro A para compararlo con un valor requerido. El programa 8.4 muestra como se utiliza esto para examinar la posición de pantalla 0,175, es decir, la esquina superior izquierda, primero imprime un cuadro negro en 0,0; después vuelve al BASIC como POINT 0,175 = 1. Esta rutina se utiliza en el programa "balloon" para apuntar (con POINT) a cada una de las posiciones del balón antes de redibujar el balón. Si la rutina vuelve con un valor de 1 entonces se llama a la rutina HIT.

org 23760  
CAPITULO 8  
PROGRAMA 4  
POINT x,y

23760 3E 02  
23762 CD 01 16  
23765 CD 6B 0D  
23768 3E 02

ld a,2  
call 5633  
call 3435  
ld a,2

23770 CD 01 16  
23773 3E 8F  
23775 D7  
23776 0E 00  
23778 06 AF  
23780 CD CE 22  
23783 CD D5 2D  
23786 FE 01  
23788 C8

call 5633  
ld a,143  
rst 16  
ld c,0  
ld b,175  
call 8910  
call 11733  
cp 1  
ret z

## Programa 8.4



# 9 LA IMPRESORA

No hay muchos programas que hagan uso de la impresora, pero puede ser conveniente tener algunas veces una copia de la pantalla (para demostrar a los amigos que usted ha logrado las puntuaciones máximas, etc.). Con esto en mente trataremos tres órdenes del BASIC: COPY, LPRINT, y LLIST y sus equivalentes en código máquina.

## COPY

Normalmente la rutina COPY pasará directamente los contenidos de las primeras 22 líneas de la pantalla a la impresora. Sin embargo, en código máquina podemos mejorar esto y especificar cuantas líneas de pantalla queremos, y la línea de comienzo; así por ejemplo, podríamos copiar sólo tres líneas empezando a partir de la línea siete.

El programa 9.1 muestra el método utilizado para copiar las 24 líneas, ya que algunos programas de juegos tienen la puntuación en las dos líneas inferiores de la pantalla y no se imprimirán con una orden directa de COPY. Puede ver que antes de llamar a 3762, el par de registros HL contienen la línea de dirección de comienzo, en nuestro ejemplo —esta es la línea 1, y el registro B guarda el número de líneas requerido  $\times 8$  ( $24 \times 8 = 192$ ).

org 23760  
CAPITULO 9  
PROGRAMA 1

COPIAR n LINEAS DE PANTALLA

HL=LINEA COMIENZO PANTALLA  
B=LINEAS  $\times 8$

EJEMPLO 24 LINEAS

23760 21 00 40	ld hl,16384
23763 06 C0	ld b,192
23765 F3	di
23766 CD B2 0E	call 3762
23769 C9	ret

### Programa 9.1

## LPRINT

Esta instrucción puede considerarse como la misma que PRINT con la única diferencia de que los caracteres de control no fun-

cionarán, por supuesto, y de que requiere que se abra un canal diferente antes de la instrucción RST 16 para enviar la salida a la impresora en vez de a la pantalla. Así, para LPRINT "A", por ejemplo, primero hay que emplear OPEN CHANNEL 3, LD A, CODE "A" y después RST 16 para que se imprima.

El programa 9.2 muestra el método anterior utilizado para imprimir (con LPRINT) una serie de caracteres. Si desea tabular su impresión con LPRINT, el número de TAB debe ponerse, con POKE, en la variable PRCC 23680 del sistema antes de la instrucción RST 16.

org 23760		23765 11 DF 5C	ld de,DATA
CAPITULO 9		23768 01 11 00	ld bc,17
PROGRAMA 2			
		LPRINT SERIES	
LPRINT			
ABRIR CANAL 3		23771 CD 3C 20	call 8252
		23774 C9	ret
		DATA	
23760 3E 03	ld a,3	defs Este es el programa 2	
23762 CD 01 16	call 5633		

## Programa 9.2

## LLIST

Aunque no encuentro ninguna aplicación en la que se requiera un listado a la mitad de un programa, he incluido éste porque es una de las órdenes de impresora y es fácil de programar. Todo lo que se necesita es llamar a 6133. El programa 9.3 muestra esto ensamblando el programa en BASIC y después listándolo con LLIST.

org 23760		30>REM org 23760
CAPITULO 9		40 REM ! CAPITULO 9; !PROGRAMA 3;!
PROGRAMA 3		50 REM ! LLIST ;!
		60 REM call 6133
LLIST		70 REM ret
		9000 REM finish
23760 CD F5 17	call 6133	9010 RANDOMIZE USR 58000
23763 C9	ret	9020 COPY
		9030 LPRINT
10 IMPRIMIR PASOS		9040 RANDOMIZE URS 23760
20 REM go		

## Programa 9.3

# 10 CONVERSION DE PROGRAMA

Si ha logrado llegar a este Capítulo comprendiendo todos los anteriores entonces le felicito —éste no le va a ocasionar ningún problema—. Si no está demasiado seguro acerca de cualquiera de las rutinas anteriores, le aconsejo que vuelva a leer el Capítulo en cuestión antes de empezar éste.

En este capítulo espero mostrar la forma de hacer un programa de juegos enteramente en código máquina utilizando un programa en BASIC como guía.

Sin duda usted tendrá su propia manera de hacer un programa en BASIC. Tal vez tiene una idea y la desarrolla en la Spectrum añadiendo rutinas conforme las va pensando, y termina con un programa completamente desestructurado, pero que funciona, y que sólo usted puede seguir. Este método es satisfactorio, pero puede darse cuenta de que si intenta arreglar el programa unas semanas después ni siquiera *usted* podrá seguirlo. Creo que el mejor método es hacer un organigrama de la idea básica y después añadir las rutinas que sean necesarias, acabando con un diagrama de flujo completo y estructurado a partir del cual puede escribirse el programa en BASIC, y tal vez mejorarse con color, sonido y UDGs. Y lo más importante, así tendrá un programa fácil de convertirse al código máquina.

El programa que convertiremos en este Capítulo se llama "CROSS" y empieza como programa para controlar a un hombre que pasa obstáculos hacia un lugar seguro. Con esta idea se hizo el organigrama. Supone que la rutina principal establece las variables, imprime las instrucciones y efectúa la visualización, y entonces un lazo examina el movimiento del hombre y su posición "nueva" de pantalla. Hay dos puntos de salida de este lazo principal:

1. Si el hombre es alcanzado ingresamos la rutina HIT y examinamos las vidas que le quedan; si no le quedan vidas el juego termina, en caso contrario volvemos a entrar en el lazo principal.
2. La rutina HOME, que también examina el número de lugares a salvo llenados, y añade arañas, o incrementa la velocidad antes de volver al lazo principal.

A partir de este diagrama de flujo se hizo la versión de

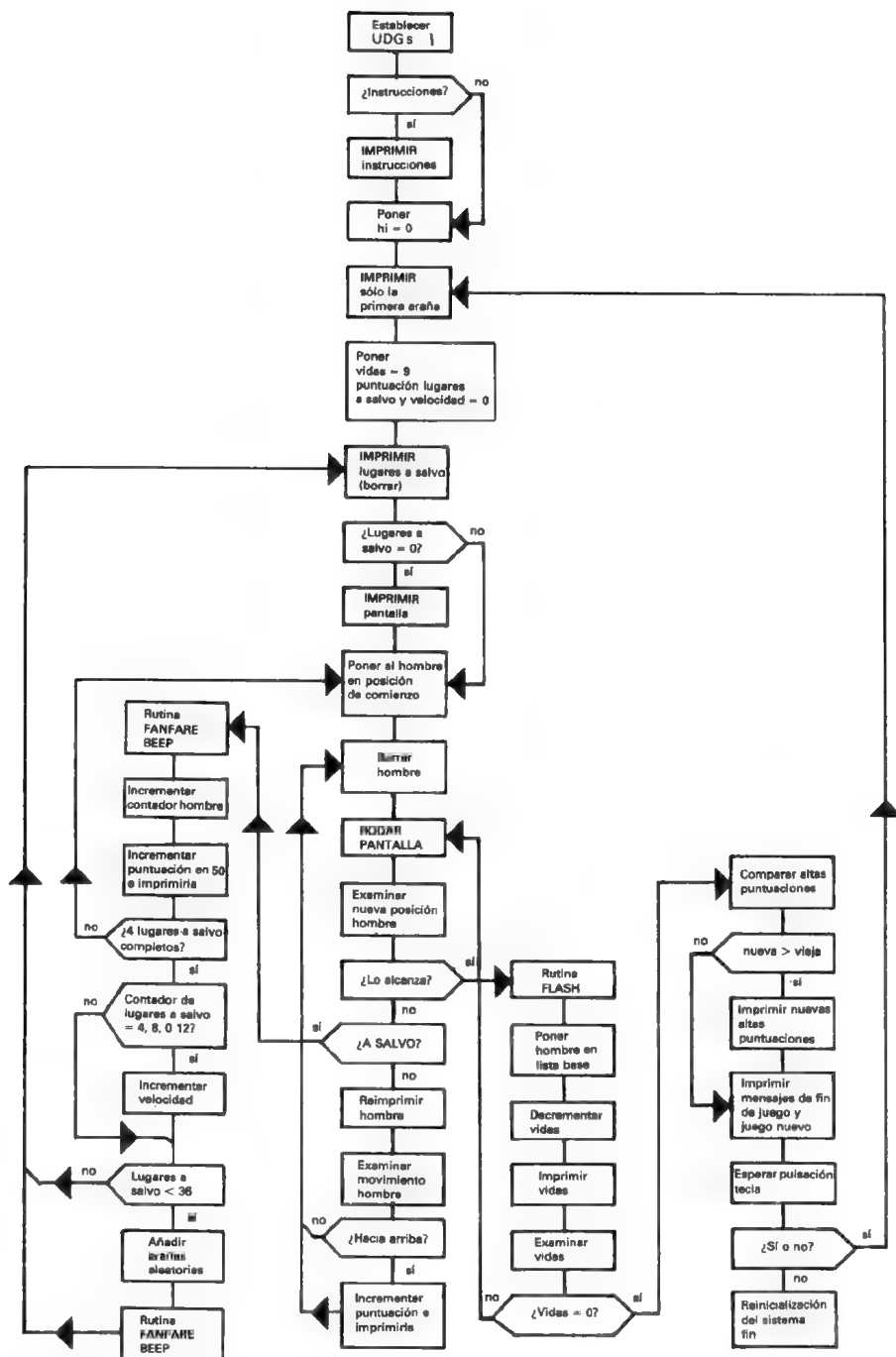


Figura 10.1 Diagrama de Flujo para el programa "Cross".

“CROSS” en BASIC. Si compara el programa actual con el diagrama de flujo podría ser capaz de reconocer todas las rutinas.

El programa, aunque es muy sencillo, se mejora mucho utilizando los UDGs, colores, y sonido. Hace un poco de trampa utilizando una rutina en código máquina para hacer rodar líneas de la pantalla a la izquierda o a la derecha un pixel cada vez, pues una versión en BASIC de esto aumentaría su lentitud de forma increíble.

Se ingresa el código máquina con POKE en la dirección de memoria correcta mediante el programa en BASIC. Esto se podría haber hecho con una instrucción LOAD código “máquina” XX, n para cargar el código directamente desde la cinta.

Antes de continuar con este Capítulo asegúrese de que comprende completamente el funcionamiento del programa, ya que ahora lo analizaremos línea por línea para convertirlo al código máquina.

```

20 CLEAR 32243: GO TO 40
25 BEEP .01, b-a
30 PRINT OVER 1; PAPER 8; INK 8; AT a, y2;
"X": RETURN
40 PRINT AT 11, 5;"por favor espere un momento"
50 FOR a=32244 TO 32494
60 READ b: POKE a, b: NEXT a
70 DATA 14, 8, 229, 17, 31, 0, 25, 126, 237,
82, 31, 6, 32, 126, 31, 119, 35, 16, 250, 225, 36,
13, 32, 234, 201
80 DATA 14, 8, 175, 229, 17, 31, 0, 237, 82,
126, 25, 23, 6, 32, 126, 23, 119, 43, 16, 250,
225, 36, 13, 32, 233, 201
90 DATA 33, 95, 64, 205, 13, 126, 33, 128, 64,
205, 244, 125, 33, 128, 64, 205, 244, 125, 33,
223, 64, 205, 13, 126, 33, 0, 72, 205, 244, 125,
33, 0, 72, 205, 244, 125, 33, 0, 72, 205, 244, 125
100 DATA 58, 121, 92, 0, 0, 0, 0, 0, 230,
2, 40, 20, 33, 64, 72, 205, 244, 125, 33, 64, 72,
205, 244, 125, 33, 64, 72, 205, 244, 125, 24, 18,
33, 95, 72, 205, 13, 126, 33, 95, 72, 205, 13,
126, 33, 95, 72, 205, 13, 126
110 DATA 33, 128, 72, 205, 244, 125, 33, 192,
72, 205, 244, 125, 33, 192, 72, 205, 244, 125,
33, 31, 80, 205, 13, 126, 33, 31, 80, 205, 13,
126, 33, 95, 80, 205, 13, 126, 201
120 DATA 33, 128, 72, 205, 244, 125, 33, 192,
72, 205, 244, 125, 33, 31, 80, 205, 13, 126, 33,
95, 80, 205, 13, 126, 201
130 DATA 33, 95, 64, 205, 13, 126, 33, 128, 64,
205, 244, 125, 33, 0, 72, 205, 244, 125, 201
140 DATA 33, 95, 64, 205, 13, 126, 33, 223, 64,
205, 13, 126, 33, 128, 72, 205, 244, 125, 33, 192,
72, 205, 244, 125, 201
150 LET a=PEEK 23675+256*PEEK 23676
160 FOR b=a TO a+167
170 READ c: POKE b, c: NEXT b
180 DATA 15, 18, 34, 127, 255, 40, 16, 128, 64,
32, 254, 254, 255, 40, 16
190 DATA 127, 127, 127, 127, 127, 255, 21, 8,
254, 254, 254, 254, 255, 255, 64, 128
200 DATA 0, 248, 196, 196, 254, 254, 40, 16,
24, 24, 36, 126, 60, 90, 165, 66
210 DATA 56, 40, 146, 124, 56, 56, 40, 108, 1,
2, 4, 127, 127, 255, 20, 8
220 DATA 240, 72, 68, 254, 255, 255, 20, 8, 0,
31, 35, 35, 127, 127, 20, 8
230 DATA 127, 127, 127, 127, 255, 255, 2, 1,
254, 254, 254, 254, 254, 255, 168, 16

```



```






600 PRINT PAPER 0; INK 4;"
610 PRINT PAPER 0; INK 7;"
620 PRINT PAPER 0; INK 6;"
630 PRINT PAPER 4;"
640 PRINT PAPER 4;"

650 PRINT PAPER 1; INK 7;" SCORE ":"AT 21,
11;" HOMBRES ":" PAPER 5; INK 0; vidas; PAPER
1; INK 7;" HISCORE "
660 LET x1=20: LET y1=16: LET x2=x1: LET
y2=y1
670 PRINT PAPER 8; INK 8; AT x1, y1; ""
680 RANDOMIZE USR 32295
690 IF SCREEN$(x2, y2)="" THEN GO TO 880
700 LET a=x2: FOR b=25 TO 35: GO SUB 25:
GO SUB 25: NEXT b
730 FOR a=x2 TO 20 STEP 2: GO SUB 25: GO
SUB 25: NEXT a
740 LET lives=lives-1: PRINT AT 21, 16; lives
750 LET x2=20
760 IF lives<>0 THEN GO TO 680
770 IF hi>score THEN GO TO 790
780 LET hi=score: PRINT AT 21, 27;hi
790 PRINT FLASH 1; PAPER 7;AT 12, 0;"
"
      GAME OVER
800 PRINT AT 14, 0;" Another game? (y)es (n)o
840 IF INKEY$="n" THEN RANDOMIZE USR 0
850 IF INKEY$<>"y" THEN GO TO 840
860 PRINT PAPER 5;AT 21, 7;" GO TO 415
880 IF x2<>0 THEN GO TO 1050
890 PRINT PAPER 8;INK 8;AT x1, y1;" ": PRINT
AT x2, y2," "
1020 IF SCREEN$(10, a+1)="" THEN
900>RESTORE 920
910 FOR a=1 TO 8: READ b, c: BEEP b, c:
NEXT a
920 DATA .1, 11, .1, 11, .8, 16, .05, 11, .05, 16,
.05, 11, .05, 16, 1, 20
930 LET home=home+1: LET
score=score+50: PRINT AT 21, 7;score
950 IF home/4<>INT (home/4) THEN GO TO
660
960 IF home=4 THEN POKE 32425, 0
970 IF home=8 THEN POKE 32450, 0
980 IF home=12 THEN POKE 32469, 0
985 IF home>36 THEN GO TO 450
990 LET a=RND*30
1000 LET a=a+1
1005 IF a>31 THEN LET a=0
1010 IF SCREEN$(10, a)="" THEN GO TO 1000
1020 IF SCREEN$(10, a+1)="" THEN GO TO
1000
1030 PRINT PAPER 4;AT 10, a; " "
1035 RESTORE 920: FOR a=1 TO 8: READ b, c:
BEEP b,c: NEXT a
1040 GO TO 450
1050 PRINT PAPER 8; INK 8;AT x2,y2," "
1060 LET x1=x2: LET y1=y2
1070 IF INKEY$<>"1" THEN GO TO 1100
1080 BEEP .001,33
1090 LET x2=x2-2: LET score=score+5: PRINT
AT 21,7;score
1100 LET y2=y2+(INKEY$="0" AND y
2<>31)-(INKEY$="9" AND y2<>0)
1110 GO TO 670

```

GRAFICOS DEFINIBLES POR EL USUARIO :

AB =   
 CDE =   
 f =   
 G =   
 HI =   
 JKL = 

M =   
 N =   
 OP =   
 QRS =   
 TU = 

## Programa "Cross basic"

org 32244  
 CAPITULO 10  
 PROGRAMA>CROSS>

[ BASIC ] CODIGO/M

### RODAR LINEA A DERECHA

DERECHA	
32244 0E 08	ld c,8
L1	
32246 E5	push hl
32247 11 1F 00	ld de,31
32250 19	add hl,de
32251 7E	ld a,(hl)
32252 ED 52	sbc hl,de
32254 1F	rra
32255 06 20	ld b,32
L2	
32257 7E	ld a,(hl)
32258 1F	rra
32259 77	ld (hl),a
32260 23	inc hl
32261 10 FA	djnz L2
32263 E1	pop hl
32264 24	inc h
32265 0D	dec c
32266 20 FA	jr nz,L1
32268 C9	ret

### RODAR LINEA A IZQUIERDA

IZQUIERDA	
32269 0E 08	ld c,8
L3	
32271 AF	xor a
32272 E5	push hl
32273 11 1E 00	ld de,31
32276 ED 52	sbc hl,de
32278 7E	ld a,(hl)
32279 19	add hl,de
32280 17	rla
32281 06 20	ld b,32
L4	
32283 7E	ld a,(hl)
32284 17	rla
32285 77	ld (hl),a
32286 2B	dec hl
32287 10 FA	djnz L4
32289 E1	pop hl
32290 24	inc h

32291 0D	dec c
32292 20 E9	jr nz,L3
32294 C9	ret

### RUTINA DE COMIENZO

HL contiene :-

Línea de comienzo para RODAR a DERECHA

Línea de final para RODAR a IZQUIERDA

### VELOCIDAD 0

32295 21 5F 40	ld hl,16479
32298 CD 0D 7E	call IZQUIERDA
32301 21 80 40	ld hl,16512
32304 CD F4 7D	call DERECHA
32307 21 80 40	ld hl,16512
32310 CD F4 7D	call DERECHA
32313 21 DF 40	ld hl,16607
32316 CD 0D 7E	call IZQUIERDA
32319 21 00 38	ld hl,18432
32322 CD F4 7D	call DERECHA
32325 21 00 48	ld hl,18432
32328 CD F4 7D	call DERECHA
32331 21 00 48	ld hl,18432
32334 CD F4 7D	call DERECHA

Mover SPIDER línea

Utiliza contador de TRAMPAS

32337 3A 79 5C	ld a,(23673)
32340 00	nop
32341 00	nop
32342 00	nop
32343 00	nop
32344 00	nop
32345 00	nop
32346 00	nop
32347 E6 02	and 2
32349 28 14	jr z,L5
32351 21 40 48	ld hl,18496
32354 CD F4 7D	call DERECHA
32357 21 40 48	ld hl,18496
32360 CD F4 7D	call RIGHT
32363 21 40 48	ld hl,18496
32366 CD F4 7D	call DERECHA
32369 18 12	jr L6
L5	
32371 21 5F 48	ld hl,18527
32374 CD 0D 7E	call IZQUIERDA
32377 21 5F 48	ld hl,18527



32380 CD 0D 7E	call IZQUIERDA	32438 21 1F 50	ld hl, 20511
32383 21 5F 48	ld hl, 18527	32441 CD 0D 7E	call IZQUIERDA
32386 CD 0D 7E	call IZQUIERDA	3244 21 5F 50	ld hl, 20575
L6		32447 CD 0D 7E	call IZQUIERDA
32389 21 80 48	ld hl, 18560	32450 C9	ret
32392 CD F4 7D	call DERECHA	VELOCIDAD 2	
32395 21 C0 48	ld hl, 18624	32451 21 5F 40	ld hl, 16479
32398 CD F4 7D	call DERECHA	32454 CD 0D 7E	call IZQUIERDA
32401 21 C0 48	ld hl, 18624	32457 21 80 40	ld hl, 16512
32404 CD F5 7D	call DERECHA	32460 CD F4 7D	call DERECHA
32407 21 1F 50	ld hl, 20511	32463 21 00 48	ld hl, 18432
32410 CD 0D 7E	call IZQUIERDA	32466 CD F4 7D	call DERECHA
32413 21 1F 50	ld hl, 20511	32469 C9	ret
32416 CD 0D 7E	call IZQUIERDA	VELOCIDAD 3	
32419 21 5F 50	ld hl, 20575	32470 21 5F 40	ld hl, 16479
32422 CD 0D 7E	call IZQUIERDA	32473 CD 0D 7E	call IZQUIERDA
32425 C9	ret	32476 21 DF 40	ld hl, 16607
VELOCIDAD 1		32479 CD 0D 7E	call IZQUIERDA
32426 21 80 48	ld hl, 18560	32482 21 80 48	ld hl, 18560
32429 CD F4 7D	call DERECHA	32485 CD F4 7D	call DERECHA
32432 21 C0 48	ld hl, 18624	32488 21 C0 48	ld hl, 18624
32435 CD F4 7D	call DERECHA	32491 CD F4 7D	call DERECHA
		32494 C9	ret

## Programa "Machine Code for Cross basic"

## Conversión en código máquina

### LINEA 20 DEL PROGRAMA EN BASIC

No será necesario convertirla a la versión en código máquina final porque no habrá necesidad de reinicializar RAMTOP para proteger el código máquina.

### LINEAS 25 Y 30 DEL PROGRAMA EN BASIC

Esta es una subrutina utilizada en la rutina HIT y en código máquina se colocará con la rutina HIT. La rutina se coloca al comienzo del programa en BASIC para reducir el tiempo que tarda la Spectrum en encontrarla. Cuando se llama a una subrutina, la Spectrum empieza desde el principio del programa en BASIC y recorre los números de línea necesarios hasta que se localiza la rutina requerida. Puede ver que si tuviese un número de línea alto llevaría más tiempo encontrarla, por lo que se incrementaría el tiempo empleado para ejecutar las líneas desde la 700 hasta la 730, que llaman a esta rutina 22 veces por lo menos.

#### **LINEA 40 DEL PROGRAMA EN BASIC**

En BASIC la Spectrum tarda varios segundos en cargar con LOAD el código máquina y establecer los UDGs, por lo que se visualiza este mensaje durante ese período. Sin embargo, en código máquina esto es instantáneo, y el mensaje es por lo tanto innecesario.

#### **LINEAS 50 A 140 DEL PROGRAMA EN BASIC**

Esta es la rutina para ingresar, con POKE, el código máquina de la "ROUTINA ROLL" en las direcciones de memoria desde 32244 a 32499, y tampoco se necesitará en código máquina, ya que estará localizada correctamente con el resto del programa en código máquina.

#### **LINEAS 150 A 270 DEL PROGRAMA EN BASIC PARA ESTABLECER UDGs**

Esta es la primera rutina que necesitaremos convertir. Por supuesto podríamos utilizar SAVE "UDG" CODE USR "a", 168: LOAD "UDG" CODE USR "a" 168, pero ello supondría tener que guardar y cargar dos grupos de rutinas en código máquina, que son el juego principal y los UDGs. Al objeto de que pueda amoldarse fácilmente a la Spectrum de 16K empezaremos el programa en código máquina en la dirección 5DC1 h(24001d). Como en el BASIC, pondremos cada byte de UDGs en forma de DATA y leeremos estos DATA transfiriendo cada número a su lugar adecuado en el área UDG de la memoria. Como verá, el método más simple es utilizar la instrucción LDIR. El par de registros DE se carga con la dirección de comienzo de UDG según lo que contenga la variable 5C7Bh(23675d) del sistema, y el par de registros HL contienen la dirección de comienzo de DATA. La rutina DATA = 5DC1h a 5E68h y la rutina en código máquina = 5E69h(24169 d) a 5E73h(24180d).

Observe que se llama a la rutina mediante RANDOMIZE USR 24169 y *no* con USR 24001 que es el comienzo de DATA.

#### **LINEA 280 DEL PROGRAMA EN BASIC**

Esta es una simple sentencia de impresión y es fácil de convertir en código máquina utilizando la rutina para imprimir con DATA de la ROM (203Ch — 8252d) que ya se estudió en el Capítulo 1. Así, DATA = 5E77h (24183d) a 5EA1h (2422 5d). Se salta a ella mediante la instrucción "JR 5EA2h" desde la direc-

ción de memoria 5E75/6h. La rutina para imprimir con DATA = 5EA2h(24226d) a 5EAFh.

#### **LINEA 290 Y 295 DEL PROGRAMA EN BASIC CODIGO MAQUINA DESDE 5EB0 A SEBE**

En esta rutina esperamos hasta que se produce una nueva pulsación de tecla y se examina la tecla para ver si es “y” (la orden HALT no es absolutamente necesaria). Si la tecla pulsada no es “y”, entonces pasamos por alto la instrucción saltando (con NZ) a 608Ah(24714d).

#### **LINEA 300 : 1 CLS DEL PROGRAMA EN BASIC CODIGO MAQUINA DESDE 5EBF a 5EC6**

Esta rutina es sencilla y se habló de ella en el Capítulo 1.

#### **LINEAS 300:2 A 330 DEL PROGRAMA EN BASIC**

De nuevo se lleva a cabo esto utilizando las rutinas de almacenamiento de DATA y las de impresión de series con PRINT. El primer grupo de DATA se guarda en 5ECA hasta 5F72 con la rutina PRINT DATA en 5F73 hasta 5F83. El segundo grupo de DATA se guarda desde 5F84 hasta 5FF9 y la rutina PRINT DATA desde 5FFA hasta 6002. Los DATA podrían haberse agrupado con una sola rutina PRINT DATA.

La rutina PAUSE 0 va desde 6003 hasta 600C; observe que el bit 5 (IY + 1) tiene que ponerse a cero desde la rutina PAUSE 0 previa. Intente sacar RES 5 (IY + 1) y “vea” el resultado.

#### **LINEA 370: 1 CLS DEL PROGRAMA EN BASIC RUTINA DESDE 600E A 6012**

Utiliza los métodos alternativos para borrar la pantalla completa ingresando ceros con POKE en todas las direcciones de la pantalla (sin afectar a los ATTRIBUTES).

#### **LINEAS 370: 2 CLS HASTA 390 DEL PROGRAMA EN BASIC**

DATA para rutina de impresión = 6016 a 6075

Rutina para imprimir datos con PRINT = 6076 a 607E.

A continuación viene de nuevo la rutina PAUSE 0.

#### **LINEA 400 DEL PROGRAMA EN BASIC**

Utilizando la rutina de establecimiento del color de BORDER y de establecimiento de atributos de color permanentes como en el Capítulo 1, podemos convertir esta línea en la rutina que va desde 608A a 609D.

La llamada CLS pone los colores en la pantalla como en el BASIC.

#### **LINEA 410 DEL PROGRAMA EN BASIC**

Esta línea establece el valor inicial de la puntuación alta. Supondremos que el valor no será mayor que 65535 y como tal utiliza los dos primeros bytes de la memoria intermedia de la impresora para guardar su valor (direcciones 5B0001-23296/7). Se puede poner a cero fácilmente con LD HL,0000h y LD (5B00),HL como se muestra en la rutina de 609C a 60A1.

#### **LINEA 420 DEL PROGRAMA EN BASIC**

DATA para impresión = 60A4 a 60C8.

Rutina para imprimir DATA con PRINT = 60C9 a 60D6

#### **LINEA 430 DEL PROGRAMA EN BASIC**

Inicializa los valores de “vidas”, “puntuación” y “lugares a salvo”. Las vidas pueden contarse con un byte, la puntuación con dos bytes, y los lugares a salvo con un byte. Emplearemos de nuevo la memoria intermedia de la impresora para almacenar esos valores. La dirección 5B02 contiene “vidas”, las 5B03/4 contienen la “puntuación”, y la 5B05 contiene “lugares a salvo”. Estos valores iniciales se almacenan utilizando la rutina de 60D7 a 60E5.

#### **LINEA 440 DEL PROGRAMA EN BASIC**

Esta línea reinicializa a cero la velocidad colocando las instrucciones RET en los lugares correctos en la subrutina LINE ROLL. Rutina: 60E6 a 60F0.

#### **LINEA 450 DEL PROGRAMA EN BASIC**

DATA para impresión = 60F4 a 6128.

Rutina para imprimir DATA con PRINT = 6129 a 6136.

#### **LINEA 455 DEL PROGRAMA EN BASIC**

Se hace una prueba para ver si los lugares a salvo < > 0; es decir, si la visualización de pantalla tiene que imprimirse o dejarse como está, borrando solamente los lugares a salvo. Necesitamos ir a por el valor almacenado en "lugares a salvo" 5B05 y comprobar que no es cero utilizando "AND A". Si esto no pone a uno la bandera de cero, entonces saltamos a la rutina de visualización PRINT. Si la bandera de cero se pone a uno, omitimos la rutina de visualización PRINT y saltamos al juego en sí.

Rutina: 6137 a 6140.

#### **LINEAS 460 A 640 DEL PROGRAMA EN BASIC**

Esta es la rutina más larga del programa, y sólo demuestra realmente lo tendioso que es imprimir una visualización.

DATA para impresión de 6141 a 63D1

Rutina para PRINT DATA 63D2 a 63DF

#### **LINEA 650 DEL PROGRAMA EN BASIC**

Se dará cuenta de que esta rutina PRINT contiene la variable "vidas" de modo que antes de que pueda usar la rutina de imprimir datos con PRINT, el valor correcto de "vidas" debe colocarse en DATA para imprimirse. Esto se logra utilizando la rutina de 63E0 a 63E9 que sólo recoge el valor "vidas" desde 5B02, le suma después 30h (48d) para convertirlo al valor ASCII correcto y finalmente lo guarda en DATA con la instrucción POKE, en la dirección 6401.

DATA para imprimir 63EA a 640F

Rutina para PRINT DATA 6410 a 641D

#### **LINEA 660 DEL PROGRAMA EN BASIC**

Esta línea establece los valores iniciales de la posición x1, y1 de impresión del hombre en la pantalla, y la posición "nueva" usada en la rutina SCREEN\$ x2, y2. De nuevo utilizaremos la memoria intermedia de la impresora para guardar estos valores.

x1 = 5B06

y1 = 5B07

x2 = 5B08

y2 = 5B09

Esto se realiza fácilmente en código máquina mediante LD HL,1014h:LD(5B06), HL : LD(5B08),HL. Por lo anterior observará que usar números hexadecimales es mucho más fácil que usar los decimales. Está bastante claro que con LD HL, 1014h contendrá 10h (16d) y L contendrá 14 (20d). Rutina de 641E a 6426.

#### **LINEA 670 DEL PROGRAMA EN BASIC**

Puesto que vamos imprimiendo un carácter cuya posición depende de las variables x1, y1, es mucho más fácil utilizar LD A,n : RST 10h que una rutina PRINT DATA. Rutina de 6427 a 6445.

#### **LINEA 680 DEL PROGRAMA EN BASIC**

Rutina de 6446 a 6448. Dejaremos la rutina ROLL en la dirección 7E27 (32295).

#### **LINEA 690 DEL PROGRAMA EN BASIC**

Utilizaremos la rutina descrita en el Capítulo 8 para encontrar SCREEN\$ (x2, y2), después desapilaremos el “valor” en los registros A,B,C,D,E. El carácter puede encontrarse entonces mediante LD A,(DE) y compararse con 20h (32). Recuerde que debemos borrar el espacio de trabajo antes de continuar con el programa. Rutina de 6449 a 645D. Si el carácter no es un espacio, saltamos a la rutina HIT que empieza en la dirección 6548h. El lazo principal del BASIC salta a la línea 880.

#### **LINEA 880 DEL PROGRAMA EN BASIC**

Aquí comprobamos que el hombre no está en la línea superior de la pantalla, es decir, en “lugar a salvo”. En código máquina obtenemos el valor x2 mediante LD A(5B08) y lo comparamos con cero. Si es cero, saltamos a la rutina de lugar seguro en la dirección 66A6. Rutina 645E a 6465. El lazo principal del BASIC salta a la línea 1050.

#### **LINEA 1050 DEL PROGRAMA EN BASIC**

Como estamos imprimiendo un carácter usando dos variables x2, y2, utilizaremos LD A,n: RST 10h (como en la línea 670). Rutina de 6466 a 6484.

#### **LINEA 1060 DEL PROGRAMA EN BASIC**

Se trata de recoger el valor almacenado en 5B08/9 y colocarlo en 5B06/7. Rutina de 6485 a 648A.

#### **LINEA 1070 DEL PROGRAMA EN BASIC**

Ahora cambiaremos el BASIC para mejorar el juego. En vez de utilizar la variable LAST KEY para leer el teclado, lo leeremos directamente empleando la orden IN A,(C) y así se podrán leer todas las teclas y se podrán hacer movimientos diagonales.

Para leer la tecla "1" necesitamos leer media fila de 1 a 5 y, como se describió en el Capítulo 5, la rutina que se requiere es 648B a 6493.

#### **LINEA 1080 DEL PROGRAMA EN BASIC**

Puesto que no es importante la duración y el tono del BEEP, no necesitamos convertirla exactamente al código máquina. Podemos usar la rutina BEEP, call 03B5(949d), en vez de utilizar la calculadora. Rutina de 6494 a 649C.

#### **LINEA 1090 DEL PROGRAMA EN BASIC**

Aquí actualizamos la posición de línea (x2) del hombre, de modo que recogemos el valor de 5B08, lo decrementamos dos veces, y volvemos a poner el nuevo valor en 5B08. Se incrementa la puntuación recogiendo el valor de 5B03/4, sumándole cinco y poniendo otra vez el resultado en 5B03/4. Rutina de 649D a 64AF. Para imprimir la puntuación utilizaremos las rutinas STACK BC y PRINT VALUE ON STACK, pero primero definiremos los parámetros de "PRINT AT". Rutina de 64B1 a 64C8.

#### **LINEA 1100 DEL PROGRAMA EN BASIC**

De nuevo, utilizando la instrucción IN A,(C) podemos leer el teclado, y actualizar el valor de y2(5B09). Rutina de 64C9 a 64F1. He añadido una rutina para examinar si se pulsa la tecla "6" para obtener la posibilidad de volver al BASIC en este punto. (Esto es necesario para guardar con SAVE el programa una vez que funcione correctamente). Rutina de 64F2 a 64F4.

Esto completa el lazo principal, pero encontrará que, si se ejecuta este programa, será tan rápido que hará casi invisible al hombre y parpadeará entre dos rodamientos de pantalla. Por

ello necesitamos un lazo de retardo para mantener quieto el hombre durante un momento, según se sugirió en el Capítulo 6. En vez de nuestro lazo normal podemos añadir un efecto de sonido de ruido de tráfico mediante la rutina de 64F5 a 650B. El JP 6427 final nos hará regresar al comienzo del lazo principal (línea 670).

## **Rutina de choque**

### **LINEA 700 DEL PROGRAMA EN BASIC**

Aquí presentamos otra variable  $a = x^2$ . Utilizaremos la dirección 5CB0 para guardar este valor (una dirección poco usada en las variables del sistema). La variable "b" del lazo FOR/NEXT puede reemplazarse por LD B,19h (25d). Este valor se coloca después en la dirección 5CB1 para utilizarse en la rutina siguiente. Rutina 6548 a 6561. La llamada GOSUB 25 se reemplaza por 650F.

### **LINEA 25 DEL PROGRAMA EN BASIC**

Esta se puede convertir fácilmente usando la calculadora para apilar .01.b —a simplemente con LD HL,(5CB0) : LDA,H : SUBL; los valores a y b se almacenan consecutivamente en 5CB0/B1. Después usamos el valor almacenado en "A" con la rutina 2D28 y la rutina BEEP 03F8. Rutina de 650F a 6521.

### **LINEA 30 DEL PROGRAMA EN BASIC**

Este es otro caso en el que usa LD A,n : RST 10h ya que hay dos variables en la rutina PRINT. Rutina de 6522 a 6547.

### **LINEA 730 DEL PROGRAMA EN BASIC**

La variable a está ya almacenada en 5CB0, por lo tanto podemos establecer un lazo tomando este valor, e incrementarlo en dos después de cada paso hasta que el valor sea 14h(20d). Rutina de 6562 a 6573.

### **LINEA 740 DEL PROGRAMA EN BASIC**

Aquí obtenemos el valor de "vidas" de la dirección 5B02, lo reducimos en uno, y volvemos a ingresar de nuevo el valor en



5B02. Para imprimir este valor establecemos los parámetros de PRINT AT, obtenemos después el valor, sumamos 30h(48d) para convertirlo al código ASCII, y lo imprimimos. Rutina de 6574 a 658E.

#### **LINEA 750 DEL PROGRAMA EN BASIC**

Rutina de 658F a 6593. Lleva consigo la reinicialización del valor x2(5B08) a 14h(20d).

#### **LINEA 760 DEL PROGRAMA EN BASIC**

Rutina de 6594 a 659A. Esta es bastante sencilla. Si “vidas” no es igual a cero, saltamos a 6446, es decir, regresamos al lazo principal.

### **Rutina de fin de juego**

Reúne la obtención del valor de alta puntuación y puntuación y comprueba cuál es el mayor. Se hace fácilmente utilizando SBC HL, DE y examinando después el estado de la bandera CARRY (acarreo). Rutina de 659B a 65A6. Si la puntuación alta es mayor que la puntuación, saltamos a 6611.

#### **LINEA 780 DEL PROGRAMA EN BASIC**

Transfiere la puntuación en 5B03/4 a la alta puntuación en 5B00/01. Rutina de 65A7 a 65AC. Para imprimir la puntuación alta utilizamos las rutinas STACK VALUE IN BC y PRINT TOP VALUE ON CALCULATOR STACK habiendo establecido primero los parámetros PRINT AT. Rutina de 65AD a 66C2.

#### **LINEAS 790 A 800 DEL PROGRAMA EN BASIC**

DATA para imprimir =65C5 a 6610.  
Rutina PRINT DATA = 6611 a 6619.

#### **LINEA 840 A 850 DEL PROGRAMA EN BASIC**

Aquí usamos la variable LAST KEY para esperar una pulsación de tecla y entrar en acción de acuerdo con esto. Rutina de

661A a 6632. Si la tecla pulsada no es “y” (79h) o “n” (6Eh) saltamos de nuevo al lazo WAIT.

#### **LINEA 860 DEL PROGRAMA EN BASIC**

Esta se lleva a cabo usando LD A,n : RST 10h seguida de JP 60C9 para volver al lazo principal reimprimiéndole la visualización, pero dejando inalterada la “puntuación alta”. Rutina de 6633 a 6655.

### **Rutina de lugar a salvo**

#### **LINEA 890 DEL PROGRAMA EN BASIC**

Aquí usamos la rutina POINT DATA pero primero ponemos en DATA los valores requeridos de x1, y1 y x2, y2.

DATA para PRINT de 669A a 66A5

SET valor de x1, y1, x2, y2 de 66A6 a 66B1

Rutina PRINT DATA de 66B2 a 66BF

#### **LINEA 900 DEL PROGRAMA EN BASIC**

Esta por supuesta es innecesaria en código máquina, ya que el comienzo de DATA no necesita reinicializarse.

#### **LINEAS 910 A 920 DEL PROGRAMA EN BASIC**

FANFARE BEEP call 666A

Se trató detalladamente esta rutina en el Capítulo 7 y por ello no necesita más comentarios. Rutina de 66C0 a 66C2.

#### **LINEAS 930 DEL PROGRAMA EN BASIC**

Otro grupo de rutinas para recoger los valores actuales de “lugares a salvo” y “puntuación” y actualizarlos; viene seguida por una rutina para imprimir la puntuación utilizando las rutinas STACT VALUE IN BC y PRINT TOP VALUE ON STACK. Rutina de actualización de 66C3 a 66D3. Rutina de impresión de puntuación de 66D4 a 66E9.

#### **LINEA 950 DEL PROGRAMA EN BASIC**

Esto es una prueba para ver si los cuatro lugares seguros están ocupados, y es más fácil de programar en código máquina que

en BASIC. Rutina de 66EA a 66F1. Puede ver que primero hacemos LD A,(5B05) es decir, contamos los "lugares a salvo" y enmascaramos entonces el valor con AND 03h. Si nos queda una respuesta distinta de cero, S2 el valor original de "lugares a salvo", no era múltiplo de cuatro. Si el resultado es distinto de cero volvemos a saltar el lazo principal con JP 641E.

#### **LINEA 960 A 980 DEL PROGRAMA EN BASIC**

Estas rutinas se utilizan para incrementar la velocidad del juego cuando "lugares a salvo" = 4,8 y 12. Esto se hace quitando las instrucciones RETURN en la subrutina ROLL, permitiendo que ciertas líneas sean cambiadas por un pixel extra: línea 960 = 66F2 a 66FF, línea 970 = 6700 a 6709, y línea 980 = 670A a 6712.

#### **LINEA 985 DEL PROGRAMA EN BASIC**

Ahora comprobamos que "lugares a salvo" no es mayor que 36. Si es así, entonces se pierde la rutina "ADD EXTRA SPIDER" ("añadir una araña extra") y volvemos al lazo principal. Restamos simplemente 37 del valor de "lugares a salvo" y si la bandera CARRY (acarreo) no está a uno (es decir lugares a salvo = 37), entonces hacemos JP NC, 6129. Rutina de 6714 a 6718.

#### **LINEA 990 DEL PROGRAMA EN BASIC**

Esta rutina fue tratada en el Capítulo 4 y no necesita más explicación, excepto anotar que en este caso utilizamos el valor SEED modificado en BC, y enmascaramos 1Fh (31d) : LD A,C : AND 1Fh; después le restamos dos y examinamos si se pone a uno la bandera de acarreo. Este método fue elegido porque no era necesario un verdadero número aleatorio. Rutina de 6719 a 6740.

#### **LINEA 1000 DEL PROGRAMA EN BASIC**

Rutina 6741.

Todo lo que se necesita es incrementar el valor de "a".

#### **LINEA 1005 DEL PROGRAMA EN BASIC**

Se hace una prueba para ver si el valor de "a" no es mayor que 31, es decir, fuera de la pantalla (10,a) se ha borrado utilizando

la rutina SCREEN\$. Esto supone colocar el valor "a" en el registro B y 0ah (10d) en el registro C, llamando a las rutinas STACK VALUE IN BC Y SCREEN\$, cambiando después los parámetros de la pila a los registros A,B,C,D,E y comprobando finalmente que SCREEN\$ (10,a) = 20h(32d). Si todo es satisfactorio, entonces obtenemos el valor final de "a" (POP AF), y se continúa. Si la rutina SCREEN\$ indica que la posición no está borrada, entonces hacemos JR 6741 y así pasamos a la siguiente posición de pantalla. Rutina de 6747 a 675E.

#### **LINEA 1020 DEL PROGRAMA EN BASIC**

Es una repetición de la rutina anterior excepto en que el valor de "a" se incrementa antes de la rutina SCREEN\$. Rutina de 675F a 6776. Observe que el valor de "a" se guarda antes de acceder a la rutina SCREEN\$ (6760 PUSH AF) para usarse en la rutina siguiente.

#### **LINEA 1030 DEL PROGRAMA EN BASIC**

Se vuelve a sacar, con POP, el valor de a, se coloca en AF y se decrementa. Este se coloca después en PRINT DATA, y se llama a la rutina PRINT. Rutina de 6777 a 6792.

#### **LINEA 1035 DEL PROGRAMA EN BASIC**

Esta es una repetición de la FANFARE BEEP y sólo requiere CALL 666A. Rutina de 6793 a 6795.

#### **LINEA 1040 DEL PROGRAMA EN BASIC**

Finalmente volvemos a saltar al lazo principal utilizando JP 6129. Rutina de 6796 a 6798.

De lo anterior puede comprobar que es realmente muy sencillo convertir un programa en BASIC a código máquina *si lo sigue paso a paso* y planea su programa con el código máquina en mente.

Un último punto, que sin duda habrá observado, es que es mejor trabajar en hexadecimal que en decimal puesto que por ejemplo, muchas de las rutinas requieren que se cargue BC con los parámetros línea/columna. Eche otro vistazo a la subrutina LEFT/RIGHT ROLL. Las direcciones de pantalla en decimal dicen muy poco, no tienen modelo aparente, pero si observa sus equivalentes en decimal verá algún tipo de modelo, obtenién-

dose una indicación más clara de lo que realmente está sucediendo.

Todo lo que se necesita ahora es que usted invente su propio programa original y brillante y empiece a convertirlo al código máquina. Le deseo buena suerte y espero que disfrute de muchas horas agradables programando sin que se le descontrolen demasiados programas.

```

org 42000 24001                                24173 21 C1 5D                                ld hl,$5dc1
                                                24176 01 AB 00                                ld bc,$00a8
                                                24179 ED B0                                  ld ir
                                                24181 18 2B                                  jr +$2b

5DC1h
defb $of $12 $22 $7f $ff $ff $28
defb $10
defb $80 $40 $20 $fe $fe $ff $28
defb $10
defb $7f $7f $7f $7f $f $ff $15
defb $08
defb $ff $fe $fe $fe $ff $ff $40
defb $80
defb $00 $f8 $c4 $c4 $fe $fe $28
defb $10
defb $18 $18 $24 $7e $3c $5a $a5
defb $42
defb $38 $28 $92 $72 $38 $38 $28
defb $6c
defb $01 $02 $4 $7f $7f $ff $14
defb $08
defb $f0 $48 $44 $fe $ff $ff $14

defb $08
defb $00 $1f $23 $23 $7f $7f $14
defb $08
defb $7f $7f $7f $7f $ff $ff $02
defb $01
defb $fe $fe $fe $fe $fe $ff $a8
defb $10
defb $10 $29 $c7 $00 $26 $00 $00
defb $00
defb $00 $44 $ff $44 $44 $ff $44

defb $00
defb $00 $22 $55 $8f $97 $a3 $a0
defb $00
defb $00 $44 $aa $f1 $e9 $c5 $05
defb $00
defb $10 $10 $10 $fe $3f $1f $0f
defb $07
defb $00 $00 $00 $00 $1e $ff $ff
defb $ff
defb $60 $7c $54 $78 $7f $ff $fe
defb $7c

defb $00 $00 $03 $02 $0f $3f $ff
defb $00
defb $06 $0c $98 $f0 $e0 $55 $ff
defb $00

5E69h
24169 ED 5B 7B 5C ld de,($5c7b)

24173 21 C1 5D                                ld hl,$5dc1
24176 01 AB 00                                ld bc,$00a8
24179 ED B0                                  ld ir
24181 18 2B                                  jr +$2b

5E77h
defb $16 $0b $03 $44 $6f $20 $79
defb $6f
defb $75 $20 $77 $61 $6e $74 $20
defb $69

defb $6e $73 $74 $72 $75 $63 $74
defb $69
defb $6f $6e $73 $3f $16 $0d $0b
defb $28
defb $79 $29 $65 $73 $16 $0f $0b
defb $2B
defb $6e $29 $6f

5EA2h
24226 3E 02                                ld a,$02
24228 CD 01 16                              call $1601
24231 11 77 5E                              ld de,$5e77
24234 01 2B 00                              ld bc,$002b
24237 CD 3C 20                              call $203c
24240 76                                    hal t
24241 FD CB 01 6E                          bit 5,(iy+$01)
24245 28 F9                                jr z,-$07
24247 3A 08 5C                              ld a,($5c08)
24250 FE 79                                cp $79

24252 C2 8A 60                                jp nz,$608a

5EBFh
24255 3E 02                                ld a,$02
24247 CD 01 16                              call $1601
24260 CD 6B 0D                              call $0d6b
24263 C3 73 5F                              jp $5f73

5ECA
defb $16 $00 $0b $4f $42 $4a $45
defb $43
defb $54 $0d $0d $54 $6f $20 $67
defb $75
defb $69 $64 $65 $20 $61 $20 $96
defb $20
defb $61 $63 $72 $6f $73 $73 $20
defb $61

```

defb \$20 \$72 \$6f \$61 \$64 \$20 \$61  
 defb \$6e  
 defb \$64 \$20 \$61 \$72 \$69 \$76 \$65  
 defb \$72  
 defb \$2c \$61 \$76 \$6f \$69 \$64 \$69  
 defb \$6e  
 defb \$67 \$20 \$90 \$91 \$20 \$92 \$93  
 defb \$94  
 defb \$20 \$95 \$20 \$a0 \$a1 \$a1 \$a2  
 defb \$20  
 defb \$a3 \$a4 \$0d \$0d \$41 \$20 \$9e  
 defb \$9f  
 defb \$20 \$70 \$61 \$74 \$72 \$6f \$6c  
 defb \$73  
 defb \$20 \$74 \$68 \$65 \$20 \$63 \$65  
 defb \$6e  
 defb \$74 \$72 \$61 \$6c \$20 \$69 \$73  
 defb \$6c  
 defb \$61 \$6e \$64 \$2e \$0d \$0d \$54  
 defb \$68  
 defb \$65 \$72 \$65 \$20 \$61 \$72 \$65

defb \$20  
 defb \$34 \$20 \$48 \$4f \$4d \$45 \$53  
 defb \$20  
 defb \$74 \$6f \$20 \$62 \$65 \$20 \$66  
 defb \$69  
 defb \$6c \$6c \$65 \$64 \$2e \$20 \$69  
 defb \$2e  
 defb \$65 \$2e \$20 \$67 \$61 \$70 \$73  
 defb \$20  
 defb \$69 \$6e \$20 \$66 \$65 \$6e \$63  
 defb \$65  
 defb \$20 \$9d \$9d \$9d \$20 \$9d \$9d  
 defb \$9d \$0d

5F73h  
 24435 3E 02                   ld a,\$02  
 24437 CD 01 16               call \$1601  
 24440 11 CA 5E               ld de,\$5eca  
 24443 01 A9 00               ld bc,\$00a9

24446 CD 3C 20               call \$203c  
 24449 C3 FA 5F               jp \$5ffa

5F84h  
 defb \$0d \$4f \$6e \$63 \$65 \$20 \$61  
 defb \$6c  
 defb \$6c \$20 \$34 \$20 \$48 \$4f \$4d  
 defb \$45  
 defb \$53 \$20 \$61 \$72 \$65 \$20 \$66  
 defb \$69  
 defb \$6c \$6c \$65 \$64 \$20 \$74 \$68  
 defb \$65  
 defb \$20 \$73 \$70 \$65 \$65 \$64 \$20  
 defb \$77  
 defb \$69 \$6c \$6c \$20 \$69 \$6e \$63  
 defb \$72  
 defb \$65 \$61 \$73 \$65 \$2c \$20 \$61  
 defb \$6e  
 defb \$20 \$65 \$78 \$74 \$72 \$61 \$20

defb \$65  
 defb \$20 \$61 \$64 \$64 \$65 \$64 \$20  
 defb \$61  
 defb \$6e \$64 \$20 \$74 \$68 \$65 \$20  
 defb \$48  
 defb \$4f \$4d \$45 \$53 \$20 \$77 \$69  
 defb \$6c  
 defb \$6c \$65 \$6d \$70 \$74 \$79 \$16  
 defb \$12  
 defb \$09 \$50 \$72 \$65 \$73 \$73 \$20  
 defb \$61  
 defb \$6e \$79 \$20 \$6b \$65 \$79

5FFAh  
 24570 11 84 5F               ld de,\$5f84  
 24573 01 76 00               ld bc,\$0076  
 24576 CD 3C 20               call \$203c  
 24579 FD CB 01 AE           res 5, (iy+\$01  
 24583 76                       halt

24584 FD CB 01 6E           bit 5,(iy+\$01  
 24588 28 F9                   jr z,-\$07

600Eh  
 24590 06 18                   ld b,\$18  
 24592 CD 44 0E               call \$0e44  
 24595 C3 76 60               jp \$6076

6016h  
 defb \$16 \$07 \$0b \$43 \$4f \$4e \$54  
 defb \$52  
 defb \$4f \$4c \$53 \$0d \$0d \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$5e \$20 \$20 \$20

defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$20 \$20  
 defb \$20

defb \$20 \$20 \$20 \$3c \$30 \$3e \$12  
 defb \$01  
 defb \$16 \$0b \$06 \$31 \$12 \$00 \$20  
 defb \$32  
 defb \$20 \$33 \$20 \$34 \$20 \$35 \$20  
 defb \$36  
 defb \$20 \$37 \$20 \$38 \$30 \$12 \$01  
 defb \$39  
 defb \$12 \$00 \$20 \$12 \$01 \$30 \$12  
 defb \$00  
 defb \$16 \$12 \$05 \$50 \$72 \$65 \$73  
 defb \$73  
 defb \$20 \$61 \$6e \$79 \$20 \$6b \$65  
 defb \$79  
 defb \$20 \$74 \$6f \$20 \$50 \$4c \$41  
 defb \$59

6076h  
24694 11 16 60  
  
24697 01 60 00  
24700 CD 3C 20  
24703 FD CB 01 AE  
24707 76  
24708 FD CB 01 6E  
24712 28 F9  
  
ld de,\$6016  
  
ld bc,\$0060  
call \$203c  
res 5, (iy+\$01  
hal t  
bit 5,(iy+\$01  
jr z,-\$07

608Ah  
24714 3E 05  
24716 CD 9B 22  
24719 3E 68  
24721 32 8D 5C  
24724 3E 02  
24726 CD 01 16  
24729 CD 6B 0D  
  
ld a,\$05  
call \$229b  
ld a,\$68  
ld (\$5c8d),a  
ld a,\$02  
call \$1601  
call \$0d6b

609Ch

24732 21 00 00  
24735 22 00 5B  
24738 18 25  
  
ld hl,\$0000  
ld (\$5b00),hl  
jr +\$25

60A4h  
defb \$11 \$04 \$16 \$0a \$00 \$20 \$20  
defb \$20  
defb \$20 \$20 \$20 \$20 \$20 \$20 \$20  
defb \$20  
defb \$20 \$20 \$20 \$20 \$20 \$9e \$9f \$20  
defb \$20

defb \$20 \$20 \$20 \$20 \$20 \$20 \$20  
defb \$20  
defb \$20 \$20 \$20 \$20 \$20 \$20

60C9h  
24777 3E 02  
  
ld a,\$02

24779 CD 01 16  
24782 11 A4 60  
24785 01 25 00  
24788 CD 3C 20  
  
call \$1601  
ld de,\$60a4  
ld bc,\$0025  
call \$203c

60D7h  
24791 3E 09  
24793 32 02 5B  
24796 21 00 00  
24799 22 03 5B  
24802 AF  
24803 32 05 5B  
  
ld a,\$09  
ld (\$5b02),a  
ld hl,\$0000  
ld (\$5b03), hl  
xor a  
ld (\$5b05),a

60E6h  
24806 3E C9  
24808 32 A9 7E  
24811 32 C2 7E  
  
ld a,\$c9  
ld (\$7ea9),a  
ld (\$7ec2),a  
  
24812 32 D5 7E  
24817 C3 29 61  
  
ld (\$7ed5),a  
jp \$6129

60F4h  
defb \$16 \$00 \$00 \$11 \$04 \$9d \$9d  
defb \$9d  
defb \$9d \$11 \$07 \$20 \$11 \$04 \$9d  
defb \$9d  
defb \$9d \$9d \$9d \$9d \$11 \$07 \$20  
defb \$11  
defb \$04 \$9d \$9d \$9d \$9d \$9d \$9d  
defb \$9d  
defb \$11 \$07 \$20 \$11 \$04 \$9d \$9d  
defb \$9d  
defb \$9d \$9d \$9d \$9d \$11 \$07 \$20  
defb \$11  
defb \$04 \$9d \$9d \$9d \$9d

6129h  
24873 3E 02  
24875 CD 01 16  
24878 11 F4 60  
24881 01 35 00  
24884 CD 3C 20  
6137h  
24887 3A 05 5B  
24890 A7  
24891 C2 1E 64  
24894 C3 D2 63  
  
ld a,\$02  
call \$1601  
ld de,\$60f4  
ld bc,\$0035  
call \$203c  
  
ld a,(\$5b05)  
and a  
jp nz,\$641 e  
jp \$63d2

6141h  
defb \$11 \$04 \$10 \$05 \$8c \$8c \$8c  
defb \$8c  
defb \$8c \$8c \$8c \$8c \$8c \$8c \$8c  
defb \$8c  
defb \$8c \$8c \$8c \$8c \$8c \$8c \$8c  
defb \$8c  
defb \$8c \$8c \$8c \$8c \$8c \$8c \$8c  
defb \$8c  
defb \$8c \$8c \$8c \$8c \$11 \$05 \$1Q  
defb \$00  
defb \$20 \$a0 \$a1 \$a1 \$a2 \$20 \$20  
defb \$20  
defb \$a0 \$a1 \$a1 \$a1 \$a2 \$20 \$20  
defb \$20  
defb \$20 \$a0 \$a1 \$a1 \$a2 \$20 \$20  
defb \$20  
defb \$a0 \$a1 \$a1 \$a2 \$20 \$20 \$20  
defb \$20  
defb \$10 \$07 \$20 \$20 \$20 \$9c \$9c  
defb \$9c  
defb \$20 \$20 \$20 \$20 \$20 \$9c \$9c  
defb \$9c  
defb \$9c \$9c \$20 \$20 \$20 \$20 \$20

defb \$9c  
 defb \$9c \$9c \$20 \$20 \$20 \$20 \$20  
 defb \$20  
 defb \$9c \$10 \$02 \$20 \$20 \$a3 \$a4  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$a3 \$a4 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$a3 \$a4 \$20 \$20  
 defb \$20  
 defb \$20 \$a3 \$a4 \$20 \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$a3 \$a4 \$20 \$10 \$07 \$9c  
 defb \$9c  
 defb \$20 \$20 \$20 \$9c \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$9c \$9c \$9c \$9c \$9c  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$20 \$20  
 defb \$9c  
 defb \$9c \$9c \$20 \$20 \$20 \$20 \$10  
 defb \$01

defb \$a1 \$a2 \$20 \$20 \$20 \$20 \$20  
 defb \$a0  
 defb \$a1 \$a1 \$a2 \$20 \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$a0 \$a1 \$a1 \$a1 \$a1 \$a2  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$a0 \$a1 \$a1

defb \$a1  
 defb \$10 \$07 \$20 \$20 \$20 \$9c \$9c  
 defb \$20  
 defb \$20 \$9c \$9c \$9c \$9c \$9c \$20  
 defb \$20  
 defb \$20 \$9c \$9c \$20 \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$9c \$9c \$9c \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$10 \$00 \$a4 \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$20 \$a3  
 defb \$a4  
 defb \$20 \$20 \$20 \$20 \$20 \$a3 \$a4  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$a3 \$a4  
 defb \$20  
 defb \$20 \$20 \$20 \$a3 \$11 \$04 \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$11  
 defb \$00  
 defb \$10 \$07 \$16 \$0d \$00 \$9d \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d  
 defb \$9d

defb \$9d \$9d \$9d \$9d \$9d \$10 \$03  
 defb \$20

defb \$90 \$91 \$20 \$95 \$20 \$20 \$90  
 defb \$91  
 defb \$20 \$20 \$92 \$93 \$92 \$93 \$94  
 defb \$20  
 defb \$95 \$20 \$95 \$20 \$20 \$20 \$92  
 defb \$93  
 defb \$94 \$20 \$20 \$20 \$20 \$20 \$20  
 defb \$10  
 defb \$07 \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$10 \$05 \$20 \$90 \$91 \$20  
 defb \$20  
 defb \$20 \$20 \$90 \$91 \$20 \$20 \$20

defb \$20  
 defb \$90 \$91 \$20 \$20 \$90 \$91 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$20 \$90

defb \$91  
 defb \$20 \$20 \$20 \$10 \$07 \$3d \$3d  
 defb \$3d  
 defb \$3d \$3d \$3d \$3d \$3d \$3d  
 defb \$3d  
 defb \$3d \$3d \$3d \$3d \$3d \$3d \$3d  
 defb \$3d  
 defb \$3d \$3d \$3d \$3d \$3d \$3d \$3d  
 defb \$3d  
 defb \$3d \$3d \$3d \$3d \$10 \$04  
 defb \$98  
 defb \$20 \$20 \$20 \$20 \$20 \$97 \$98  
 defb \$20  
 defb \$20 \$20 \$97 \$98 \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$97 \$98 \$20 \$20  
 defb \$20

defb \$20 \$20 \$20 \$20 \$20 \$20 \$ 97  
 d=fb \$10  
 defb \$07 \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$2d \$2d \$2d \$2d \$2d \$2d  
 defb \$2d  
 defb \$2d \$10 \$06 \$9a \$9b \$20 \$20  
 defb \$20  
 defb \$97 \$98 \$20 \$20 \$97 \$98 \$20  
 defb \$95  
 defb \$20 \$20 \$95 \$20 \$20 \$99 \$9a  
 defb \$9b  
 defb \$9a \$9b \$20 \$20 \$20 \$95 \$20  
 defb \$20  
 defb \$20 \$20 \$99 \$11 \$04 \$10 \$00  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d

defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d  
 defb \$9d  
 defb \$9d \$9d \$9d \$9d \$9d \$9d \$9d



deftb \$11  
 defb \$04 \$20 \$20 \$20 \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$20 \$20  
 defb \$20  
 defb \$20 \$20 \$20 \$20 \$20 \$20 \$20  
 defb \$20 \$20

63D2h  
 25554 3E 02  
 25556 CD 01 16

ld a,\$02  
 call \$1601

25559 11 41 61  
 25562 01 91 02  
 25565 CD 3C 20

ld de,\$6141  
 ld bc,\$0291  
 call \$203C

63E0h  
 25568 3A 02 5B  
 25571 C6 30  
 25573 32 01 64  
 25576 18 26

ld a,(\$5b02)  
 add a,\$30  
 ld (\$6401),a  
 jr +\$26

63EAh  
 defb \$11 \$01 \$10 \$07 \$20 \$53 \$43  
 defb \$4f  
 defb \$52 \$45 \$20 \$16 \$15 \$0b \$20  
 defb \$4d  
 defb \$45 \$4e \$20 \$11 \$05 \$10 \$00  
 defb \$39

defb \$11 \$01 \$10 \$07 \$20 \$48 \$49  
 defb \$2d  
 defb \$53 \$43 \$4f \$52 \$45 \$20

6410h  
 25616 3E 02  
 25618 CD 01 16  
 25621 11 EA 63  
 25624 01 26 00  
 25627 CD 3C 20

ld a,\$02  
 call \$1601  
 ld de,\$63ea  
 ld bc,\$0026  
 call \$203c

641Fh  
 25630 21 14 10  
 25633 22 06 5B  
 25636 22 08 5B

ld hl,\$1014  
 ld (\$5b06),hl  
 ld (\$5b08),hl

6427h  
 25639 3E 02  
 25641 CD 01 16  
 25644 3E 11  
 25646 D7  
 25647 3E 08  
 25649 D7  
 25650 3E 10  
 25652 D7  
 25653 3E 08  
 25655 D7  
 25656 3E 16  
 25658 D7  
 25659 3A 06 5B  
 25662 D7  
 25663 3A 07 5B

ld a,\$02  
 call \$1601  
 ld a,\$11  
 rst \$10  
 ld a,\$08  
 rst \$10  
 ld a,\$10  
 rst \$10  
 ld a,\$08  
 rst \$10  
 ld a,\$16  
 rst \$10  
 ld a,(\$5b06)  
 rst \$10  
 ld a,(\$5b07)

25666 D7  
 25667 3E 20  
 25669 D7

rst \$10  
 la d,\$20  
 rst \$10

6446h  
 25670 CD 27 7E

call \$7e27

6449h  
 25673 ED 4B 08 5B  
 25677 CD 38 25  
 25680 CD F1 2B  
 25683 1A  
 25684 F5  
 25685 CD BF 16  
 25688 F1  
 25689 FE 20  
 25691 C2 48 65

ld bc,(\$5b08)  
 call \$2538  
 call \$2bf1  
 ld a,(de)  
 push af  
 call \$16bf  
 pop af  
 cp \$20  
 jp nz, \$6548

645Eh

25694 3A 08 5B  
 25697 FE 00  
 25699 CA A6 66

ld a,(\$5b08)  
 cp \$00  
 jp z,\$66a6

6466h  
 25702 3E 02  
 25704 CD 01 16  
 25707 3E 11  
 25709 D7  
 25710 3E 08  
 25712 D7  
 25713 3E 10  
 25715 D7  
 25716 3E 08  
 25718 D7  
 25719 3E 16  
 25721 D7  
 25722 3A 08 5B  
 25725 D7

ld a,\$02  
 call \$1601  
 ld a,\$11  
 rst \$10  
 ld a,\$08  
 rst \$10  
 ld a,\$10  
 rst \$10  
 ld a,\$08  
 rst \$10  
 ld a,\$16  
 rst \$10  
 ld a,(\$5b08)  
 rst \$10

25726 3A 09 5B  
 25729 D7  
 25730 3E 96  
 25732 D7

ld a,(\$5b09)  
 rst \$10  
 ld a,\$96  
 rst \$10

6485h  
 25733 2A 08 5B  
 25736 22 06 5B

ld hl,(\$5b08)  
 ld (\$5b06),hl

648Bh

25739 01 FE F7  
 25472 ED 78  
 25744 CB 47  
 25746 20 1C

ld bc,\$f7fe  
 in a,(c)  
 bit 0,a  
 jr nz,+\$1c

6494h			
25748 21 32 00	ld hl,\$0032	25845 06 0A	ld b,\$0a
25751 11 05 00	ld de,\$0005	25847 11 78 00	ld de,\$0078
25754 CD B5 03	call \$03b5	25850 21 01 00	ld hl,\$0001
		25853 E5	push hl
		25854 D5	push de
		25855 C5	push bc
		25856 CD B5 03	call \$03b5
649Dh		25859 C1	pop bc
25757 3A 08 5B	ld a,(\$5b08)	25860 D1	pop de
25760 3D	dec a	25861 E1	pop hl
25761 3D	dec a	25862 7D	ld a,l
25762 32 08 5B	ld (\$5b08),a	5863 3C	inc a
25765 2A 03 5B	ld hl,(\$5b03)	25864 6F	ld l,a
25768 23	inc hl	25865 00	nop
25769 23	inc hl	25866 10 F1	djnz -\$0f
25770 23	inc hl	25868 C3 27 64	jp \$6427
25771 23	inc hl		
25772 23	inc hl		
25773 22 03 5B	ld (\$5b03),hl		
25776 00	nop	650Fh	
25777 3E 02'	ld a,\$02	25871 EF	rst \$28
25779 CD 01 16	call \$1601		
25782 3E 16	ld a,\$16		
25784 D7	rst \$10	6510h	
25785 3E 15	ld a,\$15	defb \$34 \$ea \$23 \$d7 \$0a \$3d \$38	
25787 D7	rst \$10		
25788 3E 07	ld a,\$07		
25790 D7	rst \$10		
25791 ED 4B 03 5B	ld bc,(\$5b03)	6517h	
25795 CD 2B 2D	call \$2d2b	25879 2A B0 5C	ld hl,(\$5cb0)
25798 CD E3 2D	call \$2de3	25882 7C	ld a,h
		25883 95	sub l
		25884 CD 2B 2D	call \$2d2b
		25887 CD F8 03	call \$03f8
64C9h		25890 3E 02	ld a,\$02
25801 01 FE EF	ld bc,\$effe	25892 CD 01 16	call \$1601
25804 ED 78	in a,(c)	25895 3E 15	ld a,\$15
25806 CB 47	bit 0,a	25897 D7	rst \$10
25808 20 0E	jr nz,+\$0e	25898 3E 01	ld a,\$01
25810 3A 09 5B	ld a,(\$5b09)	25900 D7	rst \$10
25813 FE 1F	cp \$1f	25901 3E 11	ld a,\$11
25815 28 01	jr z,+\$01		
25817 3C	inc a	25903 D7	rst \$10
25818 32 09 5B	ld (\$5b09),a	25904 3E 08	ld a,\$08
25821 C3 F5 64	jp \$64f5	25906 D7	rst \$10
25824 CB 4F	bit 1,a	25907 3E 10	ld a,\$10
25826 20 0E	jr nz,+\$0e	25909 D7	rst \$10
25828 3A 09 5B	ld a,(\$5b09)	25910 3E 08	ld a,\$08
25831 FE 00	cp \$00	25912 D7	rst \$10
25833 28 01	jr z,+\$01	25913 3E 16	ld a,\$16
25835 3D	dec a	25915 D7	rst \$10
		25916 3A B0 5C	ld a,(\$5cb0)
		25919 D7	rst \$10
2586 32 09 5B	ld (\$5b09),a	25920 3A 08 5B	ld a,(\$5b09)
25839 C3 F5 64	jp \$64f5	25923 D7	rst \$10
		25924 3E 96	ld a,\$96
		25926 D7	rst \$10
		25927 C9	ret
64F2h			
25842 CB 67.	bit 4,a		
25844 CB	ret z		
64F5h		6548h	
		25928 3A 08 5B	ld a,(\$5b08)

[illegible]

6633h			defb \$11 \$08 \$10 \$08 \$16 \$02 \$04
26163 3E 02	ld a,\$02		defb \$20
26165 CD 01 16	call \$1601		derb \$16 \$00 \$04 \$96
26168 3E 11	ld a,\$11		
26170 D7	rst \$10		
26171 3E 05	ld a,\$05		
26173 D7	rst \$10	66A6h	
26174 3E 16	ld a,\$16	26278 2A 06 5B	ld hl,(\$5b06)
26176 D7	rst \$10	26281 22 9F 66	ld (\$669f),hl
26177 3E 15	ld a,\$15	26284 2A 08 5B	ld hl,(\$5b08)
26179 D7	rst \$10	26287 22 A3 66	ld (\$66a3),hl
26180 3E 07	ld a,\$07	26290 3E 02	ld a,\$02
26182 D7	rst \$10	26292 CD 01 16	call \$1601
26183 3E 20	ld a,\$20	26295 11 9A 66	ld de,\$669a
26185 D7	rst \$10	26298 01 0C 00	ld bc,\$000c
26186 3E 20	ld a,\$20	26301 CD 3C 20	call \$203c
26188 D7	rst \$10	26304 CD 6A 66	call \$666a
26189 3E 20	ld a,\$20		
26191 D7	rst \$10	66C3h	
26192 3E 20	ld a,\$20	26307 3A 05 5B	ld a,(\$5b05)
26194 D7	rst \$10	26310 3C	inc a
26195 C3 C9 60	jp \$60c9	26311 32 05 5B	ld (\$5b05),a
		26314 2A 03 5B	ld hl,(\$5b03)
6656h		26317 11 32 00	ld de,\$0032
defb \$00 \$00 \$00 \$00 \$00 \$00 \$ed		26320 19	add hl,de
defb \$0b		26321 22 03 5B	ld (\$5b03),hl
defb \$ed \$0b \$f0 \$10 \$ec \$0b \$ec		26324 E5	push hl
defb \$10		26325 3E 02	ld a,\$02
defb \$ec \$0b \$ec \$10		26327 CD 01 16	call \$1601
		26330 3E 16	ld a,\$16
666Ah		26332 D7	rst \$10
26218 06 07	ld b,\$07	26333 3E 15	ld a,\$15
26220 21 5C 66	ld hl,\$665c	26335 D7	rst \$10
26223 7E	ld a,(hl)	26336 3E 07	ld a,\$07
		26338 D7	rst \$10
		26339 C1	pop bc
26225 32 78 66	ld (\$6678),a	26340 CD 2B 2D	call \$2d2b
26227 23	inc hl	26343 CD E3 2D	call \$2de3
26228 C5	push bc		
26229 E5	push hl	66EAh	
26230 EF	rst \$28	26346 3A 05 5B	ld a,(\$5b05)
defb \$34 \$ec \$4c \$cc \$cc \$cc \$38		26349 E6 03	and \$03
26238 E1	pop hl	26351 C2 1E 64	jp nz,\$641e
26239 7E	ld a,(hl)		
26240	push hl		
26241 CD 28 2D	call \$2d28	66F2h	
26244 CD F8 03	call \$03f8	26354 AF	xor a
26247 E1	pop hl	26355 3A 05 5B	ld a,(\$5b05)
26248 C1	pop bc	26358 FE 04	cp \$04
26249 23	inc hl	26360 20 06	jr nz,+\$06
26250 10 E3	djnz -\$1d	26362 AF	xor a
26252 3E 01	ld a,\$01	26363 32 A9 7E	ld (\$7ea9),a
26254 CD 28 2D	call \$2d28	26366 18 19	jr +\$19
26257 3E 14	ld a,\$14		
26259 CD 28 2D	call \$2d28		
26262 CD F8 03	call \$03f8		
26265 C9	ret		
669Ah		6700h	
		26368 FE 08	cp \$08
		26370 20 06	jr nz,+\$06
		26372 AF	xor a
		26373 32 C2 7E	ld (\$7ec2),a
		26376 18 0F	jr +\$0f

670Ah		26451 CD BF 16	call \$16bf
26378 FE 0C	cp \$0c	26454 F1	pop af
26380 20 06	jr nz,+\$06	26455 FE 20	cp \$20
26382 AF	xor a	26457 28 03	jr z,+\$03
26383 32 D5 7E	ld (\$7ed5),a	26459 F1	pop af
26386 18 05	jr +\$05	26460 18 E3	jr -\$1d
		26462 F1	pop af
6714h			
26388 D6 25	sub \$25	675Fh	
26390 D2 29 61	jp nc,\$6129	26463 3C	inc a
		26464 F5	push af
5718h			
26393 ED 4B 76 5C	ld bc,(\$5c76)	26465 47	ld b,a
26397 CD 2B 2D	call \$2d2b	26466 0F 0A	ld c,\$0a
26400 EF	rst \$28	26468 CD 38 25	call \$2538
		26471 CD F1 2B	call \$2bf1
6721h		26474 1A	ld a,(de)
defb \$a1 \$0f \$34 \$37 \$16 \$04 \$34		26475 F5	push af
defb \$80		26476 CD BF 16	call \$16bf
defb \$41 \$00 \$00 \$80 \$32 \$02 \$a1		26479 F1	pop af
defb \$03 \$31 \$38		26480 FE 20	cp \$20
		26482 28 03	jr z,+\$03
6733h		26484 F1	pop af
		26485 18 CA	jr -\$36
26419 CD A2 2D	call \$2da2	6777h	
26422 ED 43 76 5C	ld (\$5c76),bc	26487 F1	pop af
26246 79	ld a,c	26488 3D	dec a
26427 E6 1F	and \$1f	26489 32 82 67	ld (\$6782),a
26429 D6 02	sub \$02	26492 18 07	jr +\$07
26431 38 05	jr c,+\$05		
		677Eh	
		defb \$11 \$04 \$16 \$0a \$19 \$9e \$9f	
6741h			
26433 3C	inc a	6785h	
		26501 3E 02	ld a,\$02
6742h			
26434 FE 20	cp \$20	26503 CD 01 16	call \$1601
26436 20 01	jr nz,+\$01	26506 11 7E 67	ld de,\$677e
26438 AF	xor a	26509 01 07 00	ld bc,\$0007
		26512 CD 3C 20	call \$203c
6747h			
26439 F5	push af	6793h	
26440 47	ld b,a	26515 CD 6A 66	call \$666a
26441 0E 0A	ld c,\$0a		
26443 CD 38 25	call \$2538	6796h	
26446 CD F1 2B	call \$2bf1	26518 C3 29 61	jp \$6129
26449 1A	ld a,(de)		
26450 F5	push af		

## Programa "Cross (código máquina)"

## Apéndice 1

### Reemplazamiento de rutinas de la ROM

Las rutinas en ROM de la Spectrum están escritas de forma que se pueda reconocer cualquier parámetro equivocado y obtener un informe del error. Por ejemplo, si intenta ingresar `PRINT AT 24,33`; "Mensaje" entonces la rutina `PRINT` detectará que la línea 24, columna 33 está fuera de la pantalla.

Cuando se escriben programas en código máquina se puede suponer que se utilizarán siempre los parámetros correctos y así, las rutinas en ROM pueden reescribirse sin la detección de errores, etc., lo que las hace más rápidas, y pueden estar completamente bajo el control del programador.

Los seis programas siguientes muestran como pueden reescribirse `CLS.`, `ATTR.SET`, `PRINT`, `PLOT`, `UNPLOT`, `SCR$` y `POINT`. La rutina `PRINT` maneja los caracteres del 32 al 127 inclusive (aunque podría adaptarse para imprimir cualquier carácter) y realmente lo que hace es poner el carácter en la pantalla con `POKE`, por lo que es innecesario emplear `OPEN CHANNEL 2` y nos permitirá imprimir en las líneas 22 y 23. Las direcciones 23728/9 de las variables no usadas se emplean para guardar la posición actual de `PRINT`.

La rutina `SCR$` detecta los caracteres del 32 al 127 en las líneas de la 0 a la 21 como en la versión del BASIC, pero también nos permite examinar las líneas 22 y 23. Las rutinas `PLOT` y `UNPLOT` tienen los mismos parámetros que en el BASIC, con 0,0 en la esquina inferior izquierda de la línea 23. (Omitiendo las instrucciones `LDA,175` en las direcciones 2374/5 haremos que sean 0,0 en la esquina superior izquierda de la línea 1). Lo anterior se aplica también a la rutina `POINT`. La dirección a modificar es 23777. Verá que las rutinas `PLOT`, `UNPLOT` y `POINT` usan idénticas subrutinas `FIND` para encontrar el pixel a examinar; por lo tanto si usase las tres rutinas podrían compartir esta subrutina y ahorrar más memoria. Podría aumentarse todavía más la velocidad añadiendo `DI` (deshabilitar interrupciones) al comienzo de estas rutinas, o al principio de su programa, y `EI` al final (o cuando se requiera una llamada a `KEYSCAN`). Con estos programas como guía usted podría escribir una rutina `CIRCLE` (que sea más rápida que la de la Spectrum).

org 23760  
CLS (Lento)

23760 21 00 40      ld hl,16384  
L1  
23763 36 00      ld (hl),0  
23765 23      inc hl  
23766 7C      ld a,h  
23767 FE 58      cp 88  
23769 20 FB      jr nz, L1

23771 C9

CLS (Rápido)

23772 21 00 40      ld hl,16384  
23775 11 01 40      ld de,16385  
23778 01 FF17      ld bc,6143  
23781 36 00      ld (hl),0  
23783 ED B0      ldir  
23785 C9      ret

## Programa A1.1

org 23760  
LLENAR ATRIBUTOS

EJEMPLO  
INK 2 PAPER 5  
BRIGHT 1 FLASH 1

2+40+64+128=234

23760 3E EA      ld a,234  
23762 21 00 58      ld hl,22528  
23765 11 01 58      ld de,22529  
23768 01 FF 02      ld bc,767  
23771 77      ld (hl),a  
23772 ED B0      ldir  
23774 C9      ret

## Programa A1.2

org 23760  
ROUTINA IMPRIMIR SERIES  
SACA (CON POKE) CARACTERES A  
PANTALLA NO UTILIZA RST 16d

DIRECCION 23728 CONTIENE COLUMNA  
DIRECCION 23729 CONTIENE LINEA

23760 11 E4 5C      ld de,DATA  
L4  
23763 1A      ld a,(de)  
23764 CB 7F      bit 7,a  
23766 20 06      jr nz,L3  
23768 CD 02 5D      call PRINT  
23771 13      inc de  
23772 18 F5      jr L4  
L3  
23774 CB BF      res 7,a  
23776 CD 02 5D      call PRINT  
23779 C9      ret

DATA  
defs McGRAW-HILL BOOK Co.  
defs (UK) Ltd

128+CHR 46=","

defb 174

ROUTINA IMPRIMIR

OBTENER LINEA/COLUMNA EN HL

IMPRIMIR  
23810 D5      push de  
23811 2A B0 5C      ld hl,(23728)  
23814 E5      push hl  
23815 F5      push af

EXAMINAR SI SE SALE DE PANTALLA

23816 AF      xor a  
  
23817 7C      ld a,h  
23818 D6 18      sub 24  
23820 38 02      jr c,OK  
23822 CF      rst 8  
defb 4

ENCONTRAR POSICION PANTALLA

DE ACUERDO  
23824 11 00 40      ld de,16384  
23827 19      add hl,de  
23828 7C      ld a,h  
23829 E6 07      and 7  
23831 0F      rrca  
23832 0F      rrca  
23833 0F      rrca  
23834 B5      or 1  
23835 6F      ld l,a  
23836 3E F8      ld a,248  
23838 A4      and h  
23839 67      ld h,a

ENCONTRAR COMIENZO DE CAR.  
EN GENERADOR DE CAR.

23840 F1      pop af  
23841 E5      push hl  
23842 11 00 3C      ld de,15360  
23845 6F      ld l,a  
23846 26 00      ld h,0  
23848 29      add hl,hl  
23849 29      add hl,hl  
23850 29      add hl,hl  
23851 19      add hl,de  
23852 D1      pop de

## SACAR CAR. A PANTALLA (CON POKE)

```
23853 06 08      ld b,8
L1
23855 7E         ld a,(hl)
23856 12         ld (de),a

23857 23         inc hl
23858 14         inc d
23859 10 FA      djnz L1
```

## ACTUALIZAR POSICION DE PANTALLA

```
23861 E1         pop hl
23862 2C         inc l
23863 CB 6D      bit 5,l
23865 28 03      jr z,L2
23867 CB AD      res 5,l
23869 24         inc h
L2
23870 22 B0 5C   ld (23728),hl
23873 D1         pop de
23874 C9         ret
```

## Programa A1.3

org 23760

PROGRAMA PARA REEMPLAZAR  
SCREEN\$ REQUIERE M=LINEA,  
L=COLUMNA  
VUELVE CON EL REGISTRO A  
CONTENIENDO EL CODIGO CAR.  
O 0 SI CAR. NO SE ENCUENTRA

```
23760 26 0A      ld h,10
23762 2E 07      ld l,7
23764 CD D8 5C   call SCR$
23767 C9         ret
```

RUTINA SCREEN\$

SCR\$

ENCONTRAR DIRECCION DE PANTALLA  
DE CAR.

```
23768 11 00 40   ld de,16384
```

```
23771 19         add hl,de
23772 7C         ld a,h
23773 E6 07      and 7
23775 0F         rrca
23776 0F         rrca
23777 0F         rrca
23778 B5         or l
23779 6F         ld l,a
23780 3E F8      ld a,248
23782 A4         and h
23783 67         ld h,a
```

EXAMINAR CAR. CON GENE. DE CAR.

```
23784 11 00 3D   ld de,15616
L3
23787 06 08      ld b,8
23789 0E 00      ld c,0
```

EXAMINAR SI DE=16384

```
23791 CB 72      bit 6,d
23793 20 11      jr nz,NFOUND
```

```
23795 E5         push hl
L2
23796 1A         ld a,(de)
23797 BE         cp (hl)
23798 20 01      jr nz,L1
23800 0C         inc c
L1
23801 13         inc de
23802 24         inc h
23803 10 F7      djnz L2
23805 E1         pop hl
23806 CB 59      bit 3,c
23808 20 04      jr nz, ENCONTRAR
23810 18 E7      jr L3
NENCONTRAR
23812 AF         xor a
23813 C9         ret
```

CONVERTIR A CODIGO CAR.

ENCONTRAR

```
23814 B7         or a
23815 21 00 3C   ld hl,15360
23818 EB         ex de,hl
23819 ED 52      sbc hl,de
23821 06 03      ld b,3
L4
23823 CB 1C      rr h
23825 CB 1D      rr l
23827 10 FA      djnz L4
23829 7D         ld a,l
23830 3D         dec a
23831 C9         ret
```

## Programa A1.4



org 23760  
PROGRAMA PARA REEMPLAZAR  
PLOT de ROM  
REQUIERE H=y, L=x

23760 26 00 ld h,0  
23762 2E 5F ld l,95  
23764 CD D8 5C call PLOT  
23767 C9 ret  
PLOT

23768 CD DE 5C call FIND  
23771 B6 or (hl)  
23772 77 ld (hl),a  
23773 C9 ret

ENCONTRAR BIT DE PANTALLA  
VOLVER CON HL=BYTE, A=BIT

ENCONTRAR

23774 3E AF ld a,175  
23776 94 sub h

23777 67 ld h,a  
23778 11 00 40 ld de,16384  
23781 7D ld a,l  
23782 E6 07 and 7  
23784 F5 push af  
23785 7D ld a,l  
23786 E6 F8 and 248  
23788 1F rra  
23789 1F rra

23790 1F  
23791 5F  
23792 7C  
23793 E6 38  
23795 17  
23796 17  
23797 B3  
23798 5F  
23799 7C  
23800 E6 07  
23802 B2  
23803 57

23804 7C  
23805 E6 C0  
23807 1F  
23808 1F  
23809 1F  
23810 B2  
23811 57  
23812 F1  
23813 2E 80  
L1  
23815 A7  
23816 FE 00  
23818 28 05  
23820 CB 1D  
23822 3D  
23823 18 F6  
L2  
23825 7D  
23826 EB  
23827 C9

rra  
ld e,a  
ld a,h  
and 56  
rla  
rla  
or e  
ld e,a  
ld a,h  
and 7  
or d  
ld d,a

ld a,h  
and 192  
rra  
rra  
rra  
or d  
ld d,a  
pop af  
ld 1,128  
and a  
cp 0  
jr z,L2  
rr l  
dec a  
jr L1  
ld a,l  
ex de,hl  
ret

## Programa A1.5

org 23760  
PROGRAMA PARA REEMPLAZAR  
PLOT OVER 1 DE ROM  
REQUIERE H=y, L=x

23760 26 00 ld h,0  
23762 2E 5F ld l,95  
23764 CD D8 5C call PLOT OVER  
23767 C9 ret  
PLOT OVER  
23768 CD DE 5C call ENCONTRAR  
23771 AE xor (hl)  
23772 77 ld (hl),a  
23773 C9 ret

ENCONTRAR BIT PANTALLA  
VOLVER CON HL=BYTE, A=BIT

ENCONTRAR

23774 3E AF ld a,175

23776 94 sub h  
23777 67 ld h,a  
23778 11 00 40 ld de,16384  
23781 7D ld a,l

23782 E6 07  
23784 F5  
23785 7D  
23786 E6 F8  
23788 1F  
23789 1F  
23790 1F  
23791 5F  
23792 7C  
23793 E6 38  
23795 17  
23796 17  
23797 B3  
23798 5F  
23799 7C  
23800 E6 07  
23802 B2

23803 57  
23804 7C  
23805 E6 C0  
23807 1F  
23808 1F  
23809 1F  
23810 B2

and 7  
push af  
ld a,l  
and 248  
rra  
rra  
rra  
ld e,a  
ld a,h  
and 56  
rla  
rla  
or e  
ld e,a  
ld a,h  
and 7  
or d

ld d,a  
ld a,h  
and 192  
rra  
rra  
rra  
or d

23811 57	ld d,a	23820 CB 1D	rr l
23812 F1	pop af	23822 3D	dec a
23813 2e 80	ld l,128	23823 18 F6	jr L1
L1		L2	
23815 A7	and a	23825 7D	ld a,l
23816 FE 00	cp 0	23826 EB	ex de,hl
23818 28 05	jr z,L2	23827 C9	ret

## Programa A1.6

org 23760		23793 5F	ld e,a
PROGRAMA PARA		23794 7C	ld a,h
REEMPLAZAR POINT DE ROM		23795 E6 38	and 56
REQUIERE H=y, L=x		23797 17	rla
23760 26 00	ld h,0	23798 17	rla
23762 2E 5F	ld l,95	23799 B3	or e
23764 CD D8 5C	call POINT	23800 5F	ld e,a
23767 C9	ret	23801 7C	ld a,h
POINT		23802 E6 07	and 7
23768 CD E0 5C	call ENCONTRAR		
23771 A6	and (hl)	23804 B2	or d
23772 C8	ret z	23805 57	ld d,a
23773 3E 01	ld a,1	23806 7C	ld a,h
23775 C9	ret	23807 E6 C0	and 192
ENCONTRAR BIT PANTALLA		23809 1F	rra
VOLVER CON HL=BYTE, A=BIT		23810 1F	rra
		23811 1F	rra
ENCONTRAR		23812 B2	or d
		23813 57	ld d,a
		23814 F1	pop af
		23815 2E 80	ld l,128
		L1	
23776 3E AF	ld a,175	23817 A7	and a
23778 94	sub h	23818 FE 00	cp 0
23779 67	ld h,a	23820 28 05	jr z,L2
23780 11 00 40	ld de,16384	23822 CB 1D	rr l
23783 7D	ld a,l	23824 3D	dec a
23784 E6 07	and 7	23825 18 F6	jr L1
23786 F5	push af	L2	
23787 7D	ld a,l	23827 7D	ld a,l
23788 E6 F8	and 248	23828 EB	ex de,hl
23790 1F	rra		
23791 1F	rra		
23792 1F	rra	23829 C9	ret

## Programa A1.7

## Apéndice 2

### Listados del código máquina de la Spectrum (Z80)

**NOTAS:** (HL)—el número contenido en la dirección a la que apunta el par de registros HL  
 NN se ingresa con el segundo número primero, ejemplo, 4000 se ingresa 0040 (en hex.)  
 Registro A—el acumulador de propósito-general

Decimal	Bytes	Hex	Notación	Descripción
0	1	00	nop	No operación
1	3	01	ld bc,NN	Cargar BC con NN
2	1	02	ld (bc),a	Guardar A en (BC)
3	1	03	inc bc	Incrementar BC en 1
4	1	04	inc b	Incrementar B en 1
5	1	05	dec b	Decrementar B en 1
6	2	06	ld b,N	Cargar B con N
7	1	07	rlca	Rotación circular izquierda de A
8	1	08	ex af, af	Poner en activo primer AF
9	1	09	add hl,bc	BC+HL→HL
10	3	0A	ld a(bc)	Cargar A con número en localización (BC)
11	1	0B	dec bc	Decrementar BC en 1
12	1	0C	inc c	Incrementar C en 1
13	1	0D	dec c	Decrementar C en 1
14	2	0E	ld c,N	Cargar C con N
15	1	0F	rrca	Rotación circular derecha de A
16	2	10	djnz x	Decrementar B y JR si B ≠ 0, + o -x
17	3	11	ld de,NN	Cargar DE con NN
18	1	12	ld (de),a	Guardar A en (DE)
19	1	13	inc de	Incrementar DE en 1
20	1	14	inc d	Incrementar D en 1
21	1	15	dec d	Decrementar D en 1
22	2	16	ld d,N	Cargar D con N
23	1	17	rla	Rotación izquierda de A a través del acarreo
24	2	18	jr x	Salto relativo incondicional, + o -x
25	1	19	add hl,de	DE+HL→HL
26	3	1A	ld a,(de)	Cargar A con localización (DE)
27	1	1B	dec de	Decrementar DE en 1
28	1	1C	inc e	Incrementar E en 1
29	1	1D	dec e	Decrementar E en 1
30	2	1E	ld e,N	Cargar E con N
31	1	1F	rra	Rotar A a derecha a través del acarreo
32	2	20	jr nz,x	Salto relativo si no-cero, + o -x
33	3	21	ld hl,NN	Cargar HL con NN
34	3	22	ld (NN),hl	Guardar HL en localización NN
35	1	23	inc hl	Incrementar HL en 1
36	1	24	inc h	Incrementar H en 1
37	1	25	dec h	Decrementar H en 1
38	2	26	ld h,N	Cargar H con N
39	1	27	daa	Ajuste decimal de A
40	2	28	jr z, x	Salto relativo si cero + o -x
41	1	29	add hl,hl	HL+HL→HL
42	3	2A	ld hl,(NN)	Cargar HL con localización (NN)
43	1	2B	dec hl	Decrementar HL en 1
44	1	2C	inc l	Incrementar L en 1
45	1	2D	dec l	Decrementar L en 1
46	2	2E	ld l,N	Cargar L con N
47	1	2F	cpl	Complementar A (comp. en 1)
48	2	30	jr nc, x	Salto relativo si no acarreo, + o -x
49	3	31	ld sp,NN	Cargar puntero pila con NN
50	3	32	ld (NN),a	Guardar A en localización NN
51	1	33	inc sp	Incrementar SP en 1
52	1	34	inc (hl)	Incrementar (HL) en 1
53	1	35	dec(hl)	Decrementar (HL) en 1
54	2	36	ld (hl),N	Guardar N en (HL)
55	1	37	scf	Poner a 1 bandera de acarreo
56	2	38	jr c, x	Salto relativo si acarreo, + o -x
57	1	39	add hl,sp	SP+HL→HL
58	3	3A	ld a,(NN)	Cargar A con localización (NN)
59	1	3B	dec sp	Decrementar SP en 1
60	1	3C	inc a	Incrementar A en 1
61	1	3D	dec a	Decrementar A en 1

## Apéndice 2—continuación

Decimal	Bytes	Hex	Nemotécnico	Descripción
62	2	3E	Id a,N	Cargar A con N
63	1	3F	cef	Complementar bandera de acarreo
64	1	40	Id b,b	Mover B a B
65	1	41	Id b,c	Mover C a B
66	1	42	Id b,d	Mover D a B
67	1	43	Id b,e	Mover E a B
68	1	44	Id b,h	Mover H a B
69	1	45	Id b,l	Mover L a B
70	1	46	Id b,(hl)	Mover (HL) a B
71	1	47	Id b,a	Mover A a B
72	1	48	Id c,b	Mover B a C
73	1	49	Id c,c	Mover C a C
74	1	4A	Id c,d	Mover D a C
75	1	4B	Id c,e	Mover E a C
76	1	4C	Id c,h	Mover H a C
77	1	4D	Id c,l	Mover L a C
78	1	4E	Id c,(hl)	Mover (HL) a C
79	1	4F	Id c,a	Mover A a C
80	1	50	Id d,b	Mover B a D
81	1	51	Id d,c	Mover C a D
82	1	52	Id d,d	Mover D a D
83	1	53	Id d,e	Mover E a D
84	1	54	Id d,h	Mover H a D
85	1	55	Id d,l	Mover L a D
86	1	56	Id d,(hl)	Mover (HL) a D
87	1	57	Id d,a	Mover A a D
88	1	58	Id e,b	Mover B a E
89	1	59	Id e,c	Mover C a E
90	1	5A	Id e,d	Mover D a E
91	1	5B	Id e,e	Mover E a E
92	1	5C	Id e,h	Mover H a E
93	1	5D	Id e,l	Mover L a E
94	1	5E	Id e,(hl)	Mover (HL) a E
95	1	5F	Id e,a	Mover A a E
96	1	60	Id h,b	Mover B a H
97	1	61	Id h,c	Mover C a H
98	1	62	Id h,d	Mover D a H
99	1	63	Id h,e	Mover E a H
100	1	64	Id h,h	Mover H a H
101	1	65	Id h,l	Mover L a H
102	1	66	Id h,(hl)	Mover (HL) a H
103	1	67	Id h,a	Mover A a H
104	1	68	Id l,b	Mover B a L
105	1	69	Id l,c	Mover C a L
106	1	6A	Id l,d	Mover D a L
107	1	6B	Id l,e	Mover E a L
108	1	6C	Id l,h	Mover H a L
109	1	6D	Id l,l	Mover L a L
110	1	6E	Id l,(hl)	Mover (HL) a L
111	1	6F	Id l,a	Mover A a L
112	1	70	Id (hl),b	Mover B a (HL)
113	1	71	Id (hl),c	Mover C a (HL)
114	1	72	Id (hl),d	Mover D a (HL)
115	1	73	Id (hl),e	Mover E a (HL)
116	1	74	Id (hl),h	Mover H a (HL)
117	1	75	Id (hl),l	Mover L a (HL)
118	1	76	halt	PARAR
119	1	77	Id (hl),a	Mover A a (HL)
120	1	78	Id a,b	Mover B a A
121	1	79	Id a,c	Mover C a A
122	1	7A	Id a,d	Mover D a A
123	1	7B	Id a,e	Mover E a A
124	1	7C	Id a,h	Mover H a A
125	1	7D	Id a,l	Mover L a A
126	1	7E	Id a,(hl)	Mover (HL) a A
127	1	7F	Id a,a	Mover A a A
128	1	80	add a,b	B + A → A
129	1	81	add a,c	C + A → A
130	1	82	add a,d	D + A → A
131	1	83	add a,e	E + A → A
132	1	84	add a,h	H + A → A
133	1	85	add a,l	L + A → A
134	1	86	add a,(hl)	(HL) + A → A
135	1	87	add a,a	A + A → A
136	1	88	adc a,b	B + A + acarreo → A
137	1	89	adc a,c	C + A + acarreo → A
138	1	8A	adc a,d	D + A + acarreo → A

## Apéndice 2—continuación

Decimal	Bytes	Hex	Nemotécnico	Descripción
139	1	8B	adc, a,e	E + A + acarreo → A
140	1	8C	adc, a,h	H + A + acarreo → A
141	1	8D	adc, a,l	L + A + acarreo → A
142	1	8E	adc, a,(hl)	(HL) + A + acarreo → A
143	1	8F	adc, a,a	A + A + acarreo → A
144	1	90	sub b	A - B → A
145	1	91	sub c	A - C → A
146	1	92	sub d	A - D → A
147	1	93	sub e	A - E → A
148	1	94	sub h	A - H → A
149	1	95	sub l	A - L → A
150	1	96	sub (hl)	A - (HL) → A
151	1	97	sub a	A - A → A
152	1	98	sbc a,b	A - B - acarreo → A
153	1	99	sbc a,c	A - C - acarreo → A
154	1	9A	sbc a,d	A - D - acarreo → A
155	1	9B	sbc a,e	A - E - acarreo → A
156	1	9C	sbc a,h	A - H - acarreo → A
157	1	9D	sbc a,l	A - L - acarreo → A
158	1	9E	sbc a,(hl)	A - (HL) - acarreo → A
159	1	9F	sbc a,a	A - A - acarreo → A
160	1	A0	and b	A y B → A
161	1	A1	and c	A y C → A
162	1	A2	and d	A y D → A
163	1	A3	and e	A y E → A
164	1	A4	and h	A y H → A
165	1	A5	and l	A y L → A
166	1	A6	and (hl)	A y (HL) → A
167	1	A7	and a	A y A → A
168	1	A8	xor b	A exclusivo o B → A
169	1	A9	xor c	A exclusivo o C → A
170	1	AA	xor d	A exclusivo o D → A
171	1	AB	xor e	A exclusivo o E → A
172	1	AC	xor h	A exclusivo o H → A
173	1	AD	xor l	A exclusivo o L → A
174	1	AE	xor (hl)	A exclusivo o (HL) → A
175	1	AF	xor a	A exclusivo o A → A
176	1	B0	or b	A o B → A
177	1	B1	or c	A o C → A
178	1	B2	or d	A o D → A
179	1	B3	or e	A o E → A
180	1	B4	or h	A o H → A
181	1	B5	or l	A o L → A
182	1	B6	or (hl)	A o (HL) → A
183	1	B7	or a	A o A → A
184	1	B8	cp b	Comparar A:B
185	1	B9	cp c	Comparar A:C
186	1	BA	cp d	Comparar A:D
187	1	BB	cp e	Comparar A:E
188	1	BC	cp h	Comparar A:H
189	1	BD	cp l	Comparar A:L
190	1	BE	cp (hl)	Comparar A:(HL)
191	1	BF	cp a	Comparar A:A
192	1	C0	ret nz	Volver si no-cero
193	1	C1	pop bc	Sacar BC de la pila
194	3	C2	jp nz, NN	Saltar a NN si no-cero
195	3	C3	jp NN	Salto incondicional a NN
196	3	C4	call nz, NN	Llamada NN si no-cero
197	1	C5	push bc	Poner BC en la pila
198	2	C6	add a, N	A + N → A
199	1	C7	rst 0	Llamada 0000 rutina de inicio
200	1	C8	ret z	Volver si cero
201	1	C9	ret	Volver
202	3	CA	jpz, NN	Saltar a NN si cero
203	3	CB		Ver grupo especial de rutinas CB
204	3	CC	call z, NN	Llamada NN si cero
205	3	CD	call NN	Llamada NN
206	2	CE	adc a, N	A + N + acarreo → A
207	1	CF	rst 8	Llamada 0008 rutina de error
208	1	D0	ret nc	Volver si acarreo = 0
209	1	D1	pop de	Sacar DE de la pila
210	3	D2	jp nc, NN	Saltar a NN si acarreo = 0
211	2	D3	out (N), a	Sacar A al puerto N
212	3	D4	call nc, NN	Llamada a NN si acarreo = 0

## Apéndice 2—continuación

Decimal	Bytes	Hex	Nemotécnico	Descripción
213	1	D5	push de	Meter DE en la pila
214	2	D6	sub N	A ← N → A
215	1	D7	rst 16	Llamada 0010 rutina de impresión
216	1	D8	ret c	Volver si acarreo = 1
217	1	D9	exx	Poner en activo primer B-F (registros de intercambio)
218	3	DA	jp c,NN	Saltar a NN si acarreo = 1
219	2	DB	in a,(N)	Ingresar A desde el puerto N
220	3	DC	call c,NN	Llamada NN si acarreo = 1
221		DD	prefija instrucciones utilizando iy	Ver conjunto especial de rutinas DD
222	2	DE	sbc a,N	A ← N - acarreo → A
223	1	DF	rst 24	Llamada 0018 rutina de carácter (1)
224	1	E0	ret po	Volver si bandera de "overflow"/paridad = 0
225	1	E1	pop hl	Sacar HL de la pila
226	3	E2	jp po,NN	Saltar a NN si bandera de "overflow"/paridad = 0
227	1	E3	ex (sp),hl	Intercambiar (SP) y HL
228	3	E4	call po,NN	Llamada NN si bandera de "overflow"/paridad = 0
229	1	E5	push hl	Poner HL en la pila
230	2	E6	and N	A y N → A
231	1	E7	rst 32	Llamada 0020 rutina de carácter (2)
232	1	E8	ret pe	Volver si bandera de "overflow"/paridad = 1
233	1	E9	jp (hl)	Saltar a localización (HL)
234	3	EA	jp pe,NN	Saltar a NN si bandera de "overflow"/paridad = 1
235	1	EB	ex de,hl	Intercambiar DE y HL
236	3	EC	call pe,NN	Llamada NN si bandera de "overflow"/paridad = 1
237		ED		Ver conjunto especial de rutinas ED
238	2	EE	xor N	A exclusiva o N → A
239	1	EF	rst 40	Llamada 0028 rutina de calculadora
240	1	F0	ret p	Volver si bandera de signo = 0
241	1	F1	pop af	Sacar AF de la pila
242	3	F2	jp p,NN	Saltar a NN si bandera de signo = 0
243	1	F3	di	Deshabilitar interrupciones
244	3	F4	call p,NN	Llamada NN si bandera de signo = 0
245	1	F5	push af	Poner AF en pila
246	2	F6	or N	A o N = A
247	1	F7	rst 48	Llamada 0030 rutina de espacio de trabajo
248	1	F8	ret m	Volver si bandera de signo = 1
249	1	F9	ld sp,hl	Mover HL a SP
250	3	FA	jp m,NN	Saltar a NN si bandera de signo = 1
251	1	FB	ei	Habilitar interrupciones
252	3	FC	call m,NN	Llamada a NN si bandera de signo = 1
253		FD	prefija instrucciones utilizando iy	Ver conjunto especial de rutinas FD
254	2	FE	cp N	Comparar A : N
255	1	FF	rst 56	Llamada 0036

Apéndice 2—continuación

DD 09	ADD IX,BC	DD CB d 06	RLC	(IX+d)
DD 19	ADD IX,DE	DD CB d 0E	RRC	(IX+d)
DD 21 +dddd	LD IX,+dddd	DD CB d 16	RL	(IX+d)
DD 22 addr	LD (addr),IX	DD CB d 1E	RR	(IX+d)
DD 23	INC IX	DD CB d 26	SLA	(IX+d)
DD 29	ADD IX,IX	DD CB d 2E	SRA	(IX+d)
DD 2A addr	LD IX,(addr)	DD CB d 3E	SRL	(IX+d)
DD 2B	DEC IX	DD CB d 46	BIT	0,(IX+d)
DD 34 d	INC (IX+d)	DD CB d 4E	BIT	1,(IX+d)
DD 35 d	DEC (IX+d)	DD CB d 56	BIT	2,(IX+d)
DD 36 d +dd	LD (IX+d),+dd	DD CB d 5E	BIT	3,(IX+d)
DD 39	ADD IX,SP	DD CB d 66	BIT	4,(IX+d)
DD 46 d	LD B,(IX+d)	DD CB d 6E	BIT	5,(IX+d)
DD 4E d	LD C,(IX+d)	DD CB d 76	BIT	6,(IX+d)
DD 56 d	LD D,(IX+d)	DD CB d 7E	BIT	7,(IX+d)
DD 5E d	LD E,(IX+d)	DD CB d 86	RES	0,(IX+d)
DD 66 d	LD H,(IX+d)	DD CB d 8E	RES	1,(IX+d)
DD 6E d	LD L,(IX+d)	DD CB d 96	RES	2,(IX+d)
DD 70 d	LD (IX+d),B	DD CB d 9E	RS	3,(IX+d)
DD 71 d	LD (IX+d),C	DD CB d A6	RES	4,(IX+d)
DD 72 d	LD (IX+d),D	DD CB d AE	RES	5,(IX+d)
DD 73 d	LD (IX+d),E	DD CB d B6	RES	6,(IX+d)
DD 74 d	LD (IX+d),H	DD CB d BE	RES	7,(IX+d)
DD 75 d	LD (IX+d),L	DD CB d C6	SET	0,(IX+d)
DD 77 d	LD (IX+d),A	DD CB d CE	SET	1,(IX+d)
DD 7E d	LD A,(IX+d)	DD CB d D6	SET	2,(IX+d)
	ADD A,(IX+d)	DD CB d DE	SET	3,(IX+d)
DD 8E d	ADC A,(IX+d)	DD CB d E6	SET	4,(IX+d)
DD 96 d	SUB (IX+d)	DD CB d EE	SET	5,(IX+d)
DD 9E d	SBC A,(IX+d)	DD CB d F6	SET	6,(IX+d)
DD A6 d	AND (IX+d)	DD CB d FE	SET	7,(IX+d)
DD AE d	XOR (IX+d)	DD E1	POP	IX
DD B6 d	OR (IX+d)	DD E3	EX	(SP),IX
DD BE d	CP (IX+d)	DD E5	PUSH	IX
		DD E9	JP	(IX)
		DD F9	LD	SP,IX

**Instrucciones ED**

<b>ED 40</b> <b>IN</b> <b>B,(C)</b>	<b>ED 50</b> <b>IN</b> <b>D,(C)</b>	<b>ED 60</b> <b>IN</b> <b>H,(C)</b>		<b>ED A0</b> <b>LDI</b>	<b>ED B0</b> <b>LDIR</b>
<b>ED 41</b> <b>OUT</b> <b>(C),B</b>	<b>ED 51</b> <b>OUT</b> <b>(C),D</b>	<b>ED 61</b> <b>OUT</b> <b>(C),H</b>		<b>ED A1</b> <b>CPI</b>	<b>ED B1</b> <b>CPIR</b>
<b>ED 42</b> <b>SBC</b> <b>HL,BC</b>	<b>ED 52</b> <b>SBC</b> <b>HL,DE</b>	<b>ED 62</b> <b>SBC</b> <b>HL,HL</b>	<b>ED 72</b> <b>SBC</b> <b>HL,SP</b>	<b>ED A2</b> <b>INI</b>	<b>ED B2</b> <b>INIR</b>
<b>ED 43</b> <b>(addr),BC</b>	<b>ED 53</b> <b>(addr),DE</b>	<b>ED 63</b> <b>(addr),HL</b>	<b>ED 73</b> <b>(addr),SP</b>	<b>ED A3</b> <b>OUTI</b>	<b>ED B3</b> <b>OTIR</b>
<b>ED 44</b> <b>NEG</b>					
<b>ED 45</b> <b>RETN</b>					
<b>ED 46</b> <b>IM 0</b>	<b>ED 56</b> <b>IM 1</b>	<b>ED 66</b> <b>IM 2</b>			
<b>ED 47</b> <b>LD</b> <b>I,A</b>	<b>ED 57</b> <b>LD</b> <b>A,I</b>	<b>ED 67</b> <b>RRD</b>			
<b>ED 48</b> <b>IN</b> <b>C,(C)</b>	<b>ED 58</b> <b>IN</b> <b>E,(C)</b>	<b>ED 68</b> <b>IN</b> <b>L,(C)</b>	<b>ED 78</b> <b>IN</b> <b>A,(C)</b>	<b>ED A8</b> <b>LDD</b>	<b>ED B8</b> <b>LDDR</b>
<b>ED 49</b> <b>OUT</b> <b>(C),C</b>	<b>ED 59</b> <b>OUT</b> <b>(C),E</b>	<b>ED 69</b> <b>OUT</b> <b>(C),L</b>	<b>ED 79</b> <b>OUT</b> <b>(C),A</b>	<b>ED A9</b> <b>CPD</b>	<b>ED B9</b> <b>CPDR</b>
<b>ED 4A</b> <b>ADC</b> <b>HL,BC</b>	<b>ED 5A</b> <b>ADC</b> <b>HL,DE</b>	<b>ED 6A</b> <b>ADC</b> <b>HL,HL</b>	<b>ED 7A</b> <b>ADC</b> <b>HL,SP</b>	<b>ED AA</b> <b>IND</b>	<b>ED BA</b> <b>INDR</b>
<b>ED 4B</b> <b>LD</b> <b>BC,(addr)</b>	<b>ED 5B</b> <b>LD</b> <b>DE,(addr)</b>	<b>ED 6B</b> <b>LD</b> <b>HL,(addr)</b>	<b>ED 7B</b> <b>LD</b> <b>SP,(addr)</b>	<b>ED AB</b> <b>OUTD</b>	<b>ED BB</b> <b>OTRD</b>
<b>ED 4D</b> <b>RETI</b>					
<b>ED 4F</b> <b>LD</b> <b>R,A</b>	<b>ED 5F</b> <b>LD</b> <b>A,R</b>	<b>ED 6F</b> <b>RLD</b>			



## Apéndice 3

**Tabla de conversión decimal-hexadecimal**

Decimal 0-255 Hexadecimal 00-FF, byte bajo

Dec.	Hex.	Dec.	Hex.	Dec.	Hex.	2's C.	Dec.	Hex.	2's C.
0	00	64	40	128	80	-128	192	C0	-64
1	01	65	41	129	81	-127	193	C1	-63
2	02	66	42	130	82	-126	194	C2	-62
3	03	67	43	131	83	-125	195	C3	-61
4	04	68	44	132	84	-124	196	C4	-60
5	05	69	45	133	85	-123	197	C5	-59
6	06	70	46	134	86	-122	198	C6	-58
7	07	71	47	135	87	-121	199	C7	-57
8	08	72	48	136	88	-120	200	C8	-56
9	09	73	49	137	89	-119	201	C9	-55
10	0A	74	4A	138	8A	-118	202	CA	-54
11	0B	75	4B	139	8B	-117	203	CB	-53
12	0C	76	4C	140	8C	-116	204	CC	-52
13	0D	77	4D	141	8D	-115	205	CD	-51
14	0E	78	4E	142	8E	-114	206	CE	-50
15	0F	79	4F	143	8F	-113	207	CF	-49
16	10	80	50	144	90	-112	208	D0	-48
17	11	81	51	145	91	-111	209	D1	-47
18	12	82	52	146	92	-110	210	D2	-46
19	13	83	53	147	93	-109	211	D3	-45
20	14	84	54	148	94	-108	212	D4	-44
21	15	85	55	149	95	-107	213	D5	-43
22	16	86	56	150	96	-106	214	D6	-42
23	17	87	57	151	97	-105	215	D7	-41
24	18	88	58	152	98	-104	216	D8	-40
25	19	89	59	153	99	-103	217	D9	-39
26	1A	90	5A	154	9A	-102	218	DA	-38
27	1B	91	5B	155	9B	-101	219	DB	-37
28	1C	92	5C	156	9C	-100	220	DC	-36
29	1D	93	5D	157	9D	-99	221	DD	-35
30	1E	94	5E	158	9E	-98	222	DE	-34
31	1F	95	5F	159	9F	-97	223	DF	-33
32	20	96	60	160	A0	-96	224	E0	-32
33	21	97	61	161	A1	-95	225	E1	-31
34	22	98	62	162	A2	-94	226	E2	-30
35	23	99	63	163	A3	-93	227	E3	-29
36	24	100	64	164	A4	-92	228	E4	-28
37	25	101	65	165	A5	-91	229	E5	-27
38	26	102	66	166	A6	-90	230	E6	-26
39	27	103	67	167	A7	-89	231	E7	-25
40	28	104	68	168	A8	-88	232	E8	-24
41	29	105	69	169	A9	-87	233	E9	-23
42	2A	106	6A	170	AA	-86	234	EA	-22
43	2B	107	6B	171	AB	-85	235	EB	-21
44	2C	108	6C	172	AC	-84	236	EC	-20
45	2D	109	6D	173	AD	-83	237	ED	-19
46	2E	110	6E	174	AE	-82	238	EE	-18
47	2F	111	6F	175	AF	-81	239	EF	-17
48	30	112	70	176	B0	-80	240	F0	-16
49	31	113	71	177	B1	-79	241	F1	-15
50	32	114	72	178	B2	-78	242	F2	-14
51	33	115	73	179	B3	-77	243	F3	-13
52	34	116	74	180	B4	-76	244	F4	-12
53	35	117	75	181	B5	-75	245	F5	-11
54	36	118	76	182	B6	-74	246	F6	-10
55	37	119	77	183	B7	-73	247	F7	-9
56	38	120	78	184	B8	-72	248	F8	-8
57	39	121	79	185	B9	-71	249	F9	-7
58	3A	122	7A	186	BA	-70	250	FA	-6
59	3B	123	7B	187	BB	-69	251	FB	-5
60	3C	124	7C	188	BC	-68	252	FC	-4
61	3D	125	7D	189	BD	-67	253	FD	-3
62	3E	126	7E	190	BE	-66	254	FE	-2
63	3F	127	7F	191	BF	-65	255	FF	-1

Apéndice 3—continuación

Decimal 0-65 280 Hexadecimal 00-FF, byte alto

Decimal	Hex.	Decimal	Hex.	Decimal	Hex.	Decimal	Hex.
0	00	16 384	40	32 768	80	49 152	C0
256	01	16 640	41	33 024	81	49 408	C1
512	02	16 896	42	33 280	82	49 664	C2
768	03	17 152	43	33 536	83	49 920	C3
1 024	04	17 408	44	33 792	84	50 176	C4
1 280	05	17 664	45	34 048	85	50 432	C5
1 536	06	17 920	46	34 304	86	50 688	C6
1 792	07	18 176	47	34 560	87	50 944	C7
2 048	08	18 432	48	34 816	88	51 200	C8
2 304	09	18 688	49	35 072	89	51 456	C9
2 560	0A	18 944	4A	35 328	8A	51 712	CA
2 816	0B	19 200	4B	35 584	8B	51 968	CB
3 072	0C	19 456	4C	35 840	8C	52 224	CC
3 328	0D	19 712	4D	36 096	8D	52 480	CD
3 584	0E	19 968	4E	36 352	8E	52 736	CE
3 840	0F	20 224	4F	36 608	8F	52 992	CF
4 096	10	20 480	50	36 864	90	53 248	D0
4 352	11	20 736	51	37 120	91	53 504	D1
4 608	12	20 992	52	37 376	92	53 760	D2
4 864	13	21 248	53	37 632	93	54 016	D3
5 120	14	21 504	54	37 888	94	54 272	D4
5 376	15	21 760	55	38 144	95	54 528	D5
5 632	16	22 016	56	38 400	96	54 784	D6
5 888	17	22 272	57	38 656	97	55 040	D7
6 144	18	22 528	58	38 912	98	55 296	D8
6 400	19	22 784	59	39 168	99	55 552	D9
6 656	1A	23 040	5A	39 424	9A	55 808	DA
6 912	1B	23 296	5B	39 680	9B	56 064	DB
7 168	1C	23 552	5C	39 936	9C	56 320	DC
7 424	1D	23 808	5D	40 192	9D	56 576	DD
7 680	1E	24 064	5E	40 448	9E	56 832	DE
7 936	1F	24 320	5F	40 704	9F	57 088	DF
8 192	20	24 576	60	40 960	A0	57 344	E0
8 448	21	24 832	61	41 216	A1	57 600	E1
8 704	22	25 088	62	41 472	A2	57 856	E2
8 960	23	25 344	63	41 728	A3	58 112	E3
9 216	24	25 600	64	41 984	A4	58 368	E4
9 472	25	25 856	65	42 240	A5	58 624	E5
9 728	26	26 112	66	42 496	A6	58 880	E6
9 984	27	26 368	67	42 752	A7	59 136	E7
10 240	28	26 624	68	43 008	A8	59 392	E8
10 496	29	26 880	69	43 264	A9	59 648	E9
10 752	2A	27 136	6A	43 520	AA	59 904	EA
11 008	2B	27 392	6B	43 776	AB	60 160	EB
11 264	2C	27 648	6C	44 032	AC	60 416	EC
11 520	2D	27 904	6D	44 288	AD	60 672	ED
11 776	2E	28 160	6E	44 544	AE	60 928	EE
12 032	2F	28 416	6F	44 800	AF	61 184	EF
12 288	30	28 672	70	45 056	B0	61 440	F0
12 544	31	28 928	71	45 312	B1	61 696	F1
12 800	32	29 184	72	45 568	B2	61 952	F2
13 056	33	29 440	73	45 824	B3	62 208	F3
13 312	34	29 696	74	46 080	B4	62 464	F4
13 568	35	29 952	75	46 336	B5	62 720	F5
13 824	36	30 208	76	46 592	B6	62 976	F6
14 080	37	30 464	77	46 848	B7	63 232	F7
14 336	38	30 720	78	47 104	B8	63 488	F8
14 592	39	30 976	79	47 360	B9	63 744	F9
14 848	3A	31 232	7A	47 616	BA	64 000	FA
15 104	3B	31 488	7B	47 872	BB	64 256	FB
15 360	3C	31 744	7C	48 128	BC	64 512	FC
15 616	3D	32 000	7D	48 384	BD	64 768	FD
15 872	3E	32 256	7E	48 640	BE	65 024	FE
16 128	3F	32 512	7F	48 896	BF	65 280	FF

## Apéndice 4

### Literales de calculadora útiles

Literal d	Función	Acción
1	EXCHANGE	Intercambiar los dos valores superiores
3	SUBTRACT	El valor superior es restado del de abajo
4	MULTIPLY	Multiplicar los dos valores superiores
5	DIVIDE	El valor superior es dividido por el de debajo
15	ADD	Sumar los dos valores superiores
31	SIN	Seno del valor superior
32	COS	Coseno del valor superior
33	TAN	Tangente del valor superior
39	INT	Entero del valor superior
40	SQR	Raíz cuadrada del valor superior
49	DUPLICATE	Se duplica el valor superior y se pone en la parte superior de la pila
52	STK	Apilar datos almacenados en forma comprimida (ver Capítulo 7)
56	END	Terminar rutina de calculadora y volver al programa en código máquina
61	RESTACK	Reemplazar valor superior por su forma de coma flotante
160	0	Apilar 0
161	1	Apilar 1
162	$\frac{1}{2}$	Apilar $\frac{1}{2}$
163	$\pi/2$	Apilar $\pi/2$
164	10	Apilar 10

### Mapa de memoria de fichero de visualización

[illegible]

**Ejemplo:** Quinto byte de carácter en línea 13, columna 27 = 4BA0 + 1B = 4B BB h

## Apéndice 6

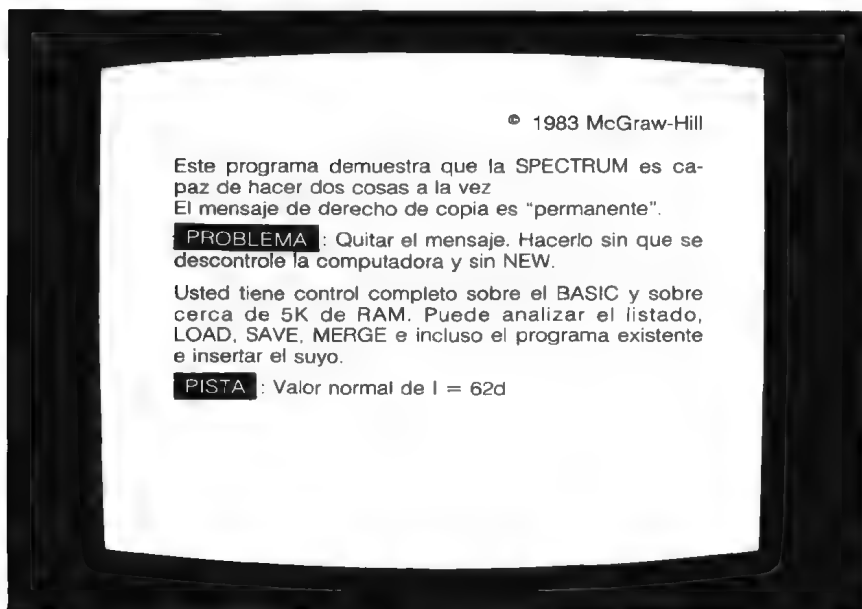
### Programa de despedida

Para aquellos de ustedes que quieran desmenuzar programas para ver como funcionan, la cinta contiene un programa en código máquina llamado PROBLEM. El objeto del programa es demostrar el hecho de que la Spectrum puede ejecutar la rutina en código máquina, a la vez que el programador tiene un control completo del BASIC.

Cuando se cargue, la computadora visualizará un mensaje de derecho de copia en la parte superior derecha de la pantalla. Este mensaje de derecho de copia permanecerá constantemente en pantalla. Su problema es averiguar cómo se consigue esto, y cambiar el mensaje sin que se des controle la Spectrum y sin utilizar NEW.

El programa está pensado de tal forma que sea imposible obtener un listado útil, y tiene varias características que hacen difícil seguir la pista del código máquina.

Si consigue modificar con éxito el mensaje y desensamblar la rutina, verá que le abre una nueva área de posibilidades de programación. Buena suerte.



```

10 RANDOMIZE USR (PEEK (PEEK 23
627+256*PEEK 23628+1) +(256*(PEEK (PEEK
23627+256*PEEK 23628+2))))

```

```

20 CLEAR 29100

```

```

30 PRINT AT 2,0; "Este programa demuestra
que la SPECTRUM es capaz de hacer dos cosas
a la vez." "El mensaje de derecho de copia es
permanente"

```

```

35 PRINT

```

```

40 PRINT "PROBLEMA: Quitar el mensaje.""

```

"Hacerlo sin que se descontrola la computadora y sin NEW"

```

50 PRINT

```

60 PRINT "usted tiene control completo sobre el BASIC y sobre cerca de 5K de RAM. Puede analizar el listado, LOAD, SAVE, MERGE e incluso el programa existente, e insertar el suyo.

```

70 PRINT "PISTA : Valor normal de I = 62d"

```

```

80 STOP

```

```

90 LOAD ""CODE : GO TO 10

```

## Programa "Problem"

## **OTRAS OBRAS DE INTERES PUBLICADAS POR OSBORNE/McGRAW-HILL**

BERMAN: *Explorando el BASIC en el COMMODORE 64.*  
CASTLEWITZ: *Introducción al VisiCalc.*  
DITLEA: *Guía de Software para microcomputadoras.*  
ERICKSON: *Telecomunicaciones con el C-64.*  
ERICKSON: *Telecomunicaciones con el MACINTOSH.*  
ETTLIN: *Manual de MBASIC.*  
ETTLIN: *Introducción al Wordstar.*  
FLAST: *1-2-3 RUN. 41 aplicaciones prácticas Lotus 1-2-3.*  
FLAST: *54 aplicaciones del VisiCalc.*  
FOX: *BASIC básico. Guía para principiantes.*  
HEILBORN: *Commodore 64. Guía del usuario.*  
HEILBORN: *Programas para ciencias e ingeniería. Edición Apple II.*  
HEILBORN: *VIC 20. Guía del usuario.*  
HOFFMAN: *Sistema operativo MS-DOS. Guía del usuario.*  
HOGAN: *Sistema operativo CP/M. Guía del usuario (2.ª ed.).*  
JEFFRIES: *Commodore 64. Pasatiempos y juegos.*  
KRUGLINSKY: *Guía a las comunicaciones del IBM/PC.*  
MUTTOLA: *Programación en lenguaje ensamblador para el Apple II.*  
OSBORNE: *Guía del comprador de sistemas de gestión.*  
OSBORNE: *Guía del ordenador personal PET/CBM.*  
POOLE: *Algunos programas de uso común en BASIC.*  
POOLE: *Algunos programas de uso común en BASIC. Edición Apple II.*  
POOLE: *Algunos programas de uso común en BASIC. Edición Atari.*  
POOLE: *Algunos programas de uso común en BASIC. Edición IBM.*  
POOLE: *Algunos programas de uso común en BASIC. Edición PET/CBM.*  
POOLE: *Algunos programas de uso común en BASIC. Edición TRS-80.*  
POOLE: *Algunos programas de uso común en Pascal.*  
POOLE: *Apple II. Guía del usuario.*  
POOLE: *Programas prácticos en BASIC.*  
POOLE: *Programas prácticos en BASIC. Edición Apple II.*  
POOLE: *Programas prácticos en BASIC. Edición IBM.*  
POOLE: *Programas prácticos en BASIC. Edición TRS-80.*  
POOLE: *Programas prácticos en Pascal.*  
SAND: *Pascal avanzado. Técnicas de programación.*  
SACHS: *El IBM/PC.*  
STEWART: *Juegos y programas educativos para C-64.*  
THOMAS: *Sistema operativo UNIX. Guía del usuario.*  
WAITE: *Introducción al procesamiento de palabras.*  
TOWNSEND: *Aplique el dBASE II.*  
FIELD: *Aplique el MacWhite y MacPaint.*

## **DISCOGUIAS PUBLICADOS POR OSBORNE/McGRAW-HILL**

GIFFOD: *Discoguía para Apple II.*  
INGRAHAM: *Discoguía para CP/M.*  
JOYANES: *Discoguía para ZX SPECTRUM (TS 2068).*  
TAYLOR: *Discoguía para Atari 400/800.*  
WILSON: *Discoguía para IBM/PC.*  
WILSON: *Discoguía para VisiCalc.*

## **OTRAS OBRAS DE INTERES PUBLICADAS POR BYTE-BOOKS/McGRAW-HILL**

ABELSON: *LOGO para Apple II.*

ABELSON: *Apple Logo.*

BOWLES: *Introducción al UCSD Pascal.*

CIARCIA: *Construya una computadora basada en el Z-80 (Guía de diseño y funcionamiento).*

CURTIS: *WORDSTAR en el IBM/PC.*

DUFF: *Introducción al MACINTOSH.*

GABY: *Gosubs. 100 subrutinas de uso común para el ZX-81 (TS 1000).*

KAMINS: *Usted y la microcomputadora. (Una introducción humanizada a la microinformática.)*

KOLVE: *Guía para seleccionar y adquirir su microcomputador.*

LEWART: *Programas de ciencias e ingeniería para microcomputadoras Sinclair ZX-81 compatibles con el ZX Spectrum.*

MORGAN: *Introducción al microprocesador 8086/8088 (16 bit).*

MULLISH: *Sinclair ZX-81 (TS 1000). Guía del usuario.*

MULLISH: *Applesoft BASIC. Guía para principiantes.*

PECKHAM: *BASIC para Apple II. Manual práctico.*

PECKHAM: *BASIC para Commodore 64. Manual práctico.*

PECKHAM: *BASIC para IBM. Manual práctico.*

PECKHAM: *BASIC para TRS-80 color. Manual práctico.*

SIKONOWIZ: *Introducción al IBM/PC.*

WATT: *Aprendiendo con LOGO.*

SKIER: *Programación en lenguaje ensamblador para VIC-20 y COMMODORE 64.*

## **OTRAS OBRAS DE INTERES PUBLICADAS POR McGRAW-HILL**

ADAMIS: *MACINTOSH. Aplicaciones de MULTIPLAN y MacPaint.*

ADAMIS: *Diccionario BASIC.*

ADAMIS: *Diccionario BASIC del IBM/PC.*

ADAMIS: *Fórmulas y programas usuales en BASIC.*

ADAMIS: *Iniciación al BASIC del IBM/PC.*

BISHOP: *ZX Spectrum (TS 2068). Teoría y proyectos de interfase.*

BUFFINGTON: *Su primera computadora: cómo comprarla y utilizarla.*

GOSLING: *Programación estructurada.*

HURLEY: *Introducción a la programación ZX-81 (TS 1000).*

HURLEY: *ZX Spectrum (TS 2068). Introducción al procesamiento de textos.*

HURLEY: *ZX Spectrum (TS 2068). Programación para jóvenes programadores.*

PHILLIPS: *Programando el Dragón. Juegos y gráficos.*

STREET: *ZX Spectrum (TS 2068). Técnicas de procesamiento de la información.*

WILLIAMS: *ZX Spectrum (TS 2068). Diseño y programación de juegos.*

WOODS: *ZX Spectrum (TS 2068). Programación en lenguaje ensamblador.*

NICHOLLS: *ZX Spectrum (TS 2068). Programación de juegos en lenguaje ensamblador.*



**OTRAS OBRAS DE INTERES PUBLICADAS POR  
McGRAW-HILL SOBRE ZX SPECTRUM (TS 2068)**

**BISHOP: ZX SPECTRUM (TS 2068).** Teoría y proyectos de interfase.

**HURLEY: ZX SPECTRUM (TS 2068).** Introducción al procesamiento de textos.

**HURLEY: ZX SPECTRUM (TS 2068).** Programación para jóvenes programadores.

**JOYANES: Disenografía para ZX SPECTRUM (TS 2068).**

**STREET: ZX SPECTRUM (TS 2068).** Técnicas de procesamiento de la información.

**WILLIAMS: ZX SPECTRUM (TS 2068).** Diseño y programación de juegos.

**WOODS: ZX SPECTRUM (TS 2068).** Programación en lenguaje ensamblador.